Université Paris Diderot – Paris 7
UFR d'Informatique

Thèse
pour l'obtention du diplôme de
Docteur de l'Université Paris Diderot, spécialité informatique

Typage et contrôle de la mobilité

présentée et soutenue publiquement par

Samuel Hym

le 1er décembre 2006

devant le jury composé de

M. Pierre-Louis	Curien	président
M. Vincent	Danos	directeur
M. Jean	Goubault-Larrecq	rapporteur
M. Matthew	Hennessy	directeur
M. Jean-Jacques	Lévy	
M. Robin	Milner	rapporteur

Merci

à Matthew Hennessy, qui a relevé le défi de diriger cette thèse à distance,

à Vincent Danos, qui l'a rendue possible,

à Jean Goubault-Larrecq et Robin Milner qui ont accepté d'être rapporteurs, ce dont je suis extrêmement honoré,

à Pierre-Louis Curien et Jean-Jacques Lévy qui ont accepté de faire partie du jury,

à Emmanuel Beffara, pour ses éclaircissements d'idées dans les premiers stades de la rédaction,

à Alexandre Miquel, pour les derniers stades de la rédaction et sa cellule de soutien psychologique,

à Francesco Zappa Nardelli, entre mille choses pour sa sagesse : « Quoi qu'il arrive, finis ta thèse en trois ans »,

à Petite Fille, pour sa quête d'articles au péril de sa vertu,

à l'exceptionnelle équipe de relecteurs[1] : Samuel Mimram, Nicolas Tabareau, Claire David, Raphaëlle Nollez-Goldbach, Christine Tasson, Séverine Maingaud, Bernard Hym, Caroline Priou, Marie Albenque, Sylvain Lebresne,

à Odile Ainardi et Michèle Wasse, sans qui la vie serait un cauchemar administratif,

à Fabien Tarissan, pour son partage des moments difficiles de fin de thèse,

à Frédéric Peschanski, mon co-auteur,

aux membres de mes laboratoires d'attache pour la joyeuse atmosphère de travail qu'ils distillent,

à Angélique et Emmanuel Hym pour leur aide « logistique »,

et enfin à Tom Waits, pour l'atmosphère quasi-quotidienne de ma rédaction.

[1] Sans ordre. Enfin... J'ai évité le travers professoral (50 points par erreur scientifique, 3 points par erreur grammaticale, etc. jusqu'aux -5 points par erreur injustement relevée, par exemple) pour tomber dans l'excès informaticien : les noms sont triés par MD5 croissants...

Table des matières

Introduction

La *mobilité* de programmes prend vie sous nos yeux telle la créature du docteur Frankenstein [She18] : parce qu'elle est simplement rendue *possible* par l'omniprésence et l'interconnexion des ordinateurs et parce qu'elle reste largement *non maîtrisée*. Cette mobilité prend différentes formes : de situations très simples, comme lorsqu'un navigateur exécute une appliquette[2] qui ne nécessite aucun échange d'informations avec qui que ce soit, à des cas plus complexes, par exemple lorsque plusieurs services web, eux-mêmes constitués d'une multitude d'ordinateurs, collaborent à un même calcul. Mais les langages de programmation actuels, de C à Haskell en passant par Java et ML, sont essentiellement conçus pour le calcul local, avec un support faible voire inexistant pour l'écriture d'applications réparties. Cette thèse s'inscrit dans le large effort de recherche fondamentale sur les langages de programmation visant à mieux comprendre ces systèmes dont la structure est si complexe. Cette recherche s'oriente suivant deux grands axes : la conception de langages possédant des primitives de plus haut niveau et le développement de méthodes de vérification de correction des programmes répartis. Cette thèse suit ce second axe et porte pour cela sur un *calcul*, au sens de formalisme mathématique sur lequel raisonner, permettant de décrire la répartition du *calcul*, au sens de l'activité effectuée par une machine numérique[3].

Plus précisément, cette thèse concerne les systèmes de typage pour le Dπ-calcul, une version répartie du π-calcul. Ses résultats principaux sont la construction d'un système de typage en présence de types récursifs et la mise en place d'une théorie de typage cohérente quand les migrations peuvent être contrôlées.

Sémantique de la concurrence

La sémantique, en tant qu'étude formelle des langages de programmation, cherche à apporter une signification mathématique précise à la suite de symboles qui compose, syntaxiquement, un programme. Cette définition précise permet de lever toute ambiguïté et, ainsi, de raisonner sur les programmes.

L'étude sémantique du λ-calcul [Chu41, Bar84] a permis de se concentrer sur un formalisme simple pour raisonner sur les programmes séquentiels, c'est-à-dire les programmes constitués d'une suite d'instructions à effectuer les unes après les

[2] traduction relativement répandue du terme anglais *applets*, pour ce qu'elle vaut.

[3] L'anglais distingue nettement les deux significations du mot français *calcul* par les deux termes *calculus* et *computation*. L'ambiguïté du terme français sera levée par le contexte dans cette thèse.

autres. Ce formalisme définit de façon élémentaire ce qu'est une fonction. Profitant de la sobriété du λ-calcul, on peut définir de diverses façons la sémantique précise du calcul et obtenir une caractérisation complète de l'équivalence naturelle grâce aux arbres de Böhm correspondant aux λ-termes.

Cependant, le λ-calcul est inadapté pour décrire des programmes *concurrents*, mettant en jeu plusieurs composantes qui s'exécutent quasi-simultanément et qui collaborent. En particulier, les objets exprimés par le λ-calcul sont essentiellement des fonctions mathématiques calculables : leur comportement se résume pratiquement à un résultat produit en utilisant ou non les arguments fournis. Au contraire, le comportement d'un programme concurrent consiste fondamentalement dans le schéma des interactions qu'il peut engager.

C'est pourquoi Robin Milner a introduit dans les années 70 le Calculus of Communicating Systems (CCS, [Mil89]). Ce calcul décrit des entités, appelées *processus*, capables d'évoluer indépendamment du reste de l'univers ou bien de se synchroniser avec d'autres processus. Afin de décrire des processus complexes, ce calcul repose sur une structure algébrique : si P et Q sont deux processus,

$$P \mid Q$$

désigne leur *composition concurrente* ou *composition parallèle*. Dans [Mil91], Robin Milner définit la sémantique d'un processus par les *réductions* qu'il peut effectuer[4]. Le point clef de cette définition de la sémantique est dans l'ensemble des réductions que peut effectuer une composition parallèle de processus. Ainsi la règle

$$\frac{P \longrightarrow P'}{P \mid Q \longrightarrow P' \mid Q}$$

et sa symétrique

$$\frac{Q \longrightarrow Q'}{P \mid Q \longrightarrow P \mid Q'}$$

signifient que la composition des processus P et Q peut se réduire dès que l'un des deux effectue une réduction. On voit sur ces deux règles que la composition parallèle *entrelace* les réductions des processus qu'elle compose. Mais la notion de composition parallèle joue surtout un rôle central au cours des synchronisations. Ces synchronisations supposent que deux processus soient prêts à interagir sur un *canal* partagé ; elles sont asymétriques : un des deux processus *émet* un *message* que l'autre *reçoit*. La réduction correspondante prend la forme

$$a!\, P \mid a?\, Q \longrightarrow P \mid Q$$

où $a!\, P$ est le processus apte à émettre un message sur le canal a puis à continuer comme P et où $a?\, Q$ est le processus qui reçoit.

Pour effectivement raisonner sur les processus, il est indispensable de définir quand deux processus sont « égaux ». CCS fournit pour cela une notion d'*équivalence observationnelle* : on ne veut en effet distinguer deux processus que lorsque leur différence est *observable* au cours d'une suite d'interactions avec eux, sans quoi rien ne justifie de les différencier. Malheureusement plusieurs équivalences coexistent. Elles reposent cependant toutes sur le mécanisme de la

[4] [Mil91] porte en fait sur le π-calcul. La sémantique par réductions de CCS est définie, par exemple, dans [Mil99].

bisimulation, introduite par David Park dans [Par81] pour comparer des automates. La bisimulation stipule que deux processus ne peuvent être équivalents que si, lorsque l'un des deux *évolue*, l'autre peut également *évoluer* de sorte que les processus résultants soient encore équivalents. Cette notion est donc particulièrement adaptée à la comparaison de processus dont le déroulement est potentiellement infini. En donnant à « évoluer » le sens de « se réduire », ce mécanisme est l'une des trois notions fondamentales intervenant dans la définition de la *congruence barbue* introduite par Robin Milner et Davide Sangiorgi dans [MS92]. La deuxième notion identifie une capacité d'observation minimale, appelée *barbe*, permettant de détecter les aptitudes du processus à interagir. Ainsi le processus $a?\,P$ exhibe une barbe $a?$ permettant de le différencier du processus $b?\,P$. Enfin, la troisième notion permettant de définir l'équivalence consiste, comme le nom de congruence l'indique, en ce que deux processus équivalents doivent rester observationnellement indistinguables quel que soit le contexte dans lequel ils sont placés. La notion de congruence barbue occupe une place centrale dans le travail présenté dans cette thèse et sera donc présentée plus en détails par la suite.

La définition de bisimulation s'applique aussi en donnant à « évoluer » un sens plus vaste que simplement « se réduire ». On peut ainsi identifier les *actions* qu'un processus est apte à faire par des *transitions étiquetées* et considérer les actions comme évolutions. Ainsi les transitions étiquetées prennent la forme

$$a?\,P \xrightarrow{a?} P$$

pour signifier que le processus $a?\,P$ est capable d'effectuer l'action $a?$ pour devenir P. Cette action caractérise l'interaction que peut engager le processus avec son contexte. Deux processus sont alors *bisimilaires* à la condition que toute action effectuée par l'un puisse aussi être effectuée par l'autre et que les deux processus obtenus après cette action soient encore bisimilaires. C'est cette équivalence que l'on appelle *bisimilarité* et qui joue, elle aussi, un rôle central dans cette thèse.

Ces équivalences engendrent deux variantes chacune, suivant la précision avec laquelle l'observateur est apte à détecter le fait qu'un processus se réduit. Dans les variantes dites *fortes*, chaque évolution d'un processus doit correspondre à exactement une évolution de tout processus équivalent. Les variantes dites *faibles* autorisent le processus qui doit être équivalent à effectuer une série de réductions internes invisibles supplémentaires. Ce sont ces variantes faibles qui présentent le plus grand intérêt comme notion d'équivalence car elles permettent de relier un processus simple représentant une spécification (le comportement attendu d'une machine, par exemple) à celui représentant l'implémentation (la façon dont la machine fonctionne effectivement). L'implémentation d'une machine à café, par exemple, devra en effet décomposer la production d'une tasse de café en plusieurs étapes, notamment le versement du sucre puis celui du café proprement dit. Ces différentes étapes pourront alors être abstraites, pour ne considérer que l'essentiel de ce comportement, ce qui est réellement observable par l'utilisateur, à savoir la production d'une tasse de café sucré[5].

La question naturelle qui naît de la cohabitation de ces deux équivalences est de savoir comment elles se comparent. La congruence barbue se définit de façon assez naturelle car les critères qui la composent sont intuitifs. Malheureusement,

[5] enfin, si on peut appeler ça un café...

la preuve que des processus sont congrus suppose une quantification sur tous les contextes pour vérifier que ces deux processus y restent équivalents. Une telle preuve est donc, au mieux, fort laborieuse. La bisimilarité dispose quant à elle des avantages inverses. En effet, la condition qui la définit n'est pas à proprement parler intuitive, mais elle fournit une technique de preuve bien plus commode. L'idéal serait donc d'arriver à prouver que ces deux équivalences coïncident. Dans le cadre de CCS, une petite condition supplémentaire sur les processus, simple à vérifier, permet d'obtenir cette coïncidence. Dans la suite de cette thèse, on prouvera que ces deux équivalences coïncident dans Dπ.

Formalismes pour la mobilité

Malgré toutes les propriétés qui peuvent déjà être étudiées dans le cadre de CCS, ce calcul impose une structure de communication figée : l'ensemble des noms partagés par deux processus ne peut en effet pas varier au cours de l'exécution. L'ensemble des comportements que le calcul est à même de modéliser est donc relativement restreint. Le π-calcul [Mil99, SW01], introduit par Robin Milner, Joachim Parrow et David Walker dans [MPW92], résout cette difficulté et franchit une grande étape en permettant de créer de nouveaux canaux et de communiquer des noms de canaux sur un canal. Ce saut conceptuel aboutit alors à un calcul de processus dans lequel de nouveaux liens entre processus peuvent se faire et se défaire dynamiquement. La règle de réduction correspondant à la communication y prend donc la forme

$$a\,!\,\langle V\rangle\,P_1 \mid a\,?\,(X)\,P_2 \longrightarrow P_1 \mid P_2\{^V/_X\}$$

La démarche de développement de la sémantique du π-calcul, avec une sémantique par réductions et deux approches différentes de la notion d'équivalence, est très similaire à celle de CCS. Les équivalences y sont cependant nettement plus complexes, car elles doivent prendre en compte les échanges de noms ; les comparaisons entre les équivalences sont aussi plus difficiles à établir.

Mais tous les modèles évoqués jusqu'à présent ignorent la répartition spatiale du calcul. On voudrait pourtant pouvoir représenter fidèlement cette répartition. Par exemple, si deux machines effectuent leurs propres calculs mais échangent certaines de leurs conclusions, et que l'on essaye de modéliser cette situation en π-calcul, rien ne pourra différencier les processus représentant les calculs se déroulant sur une machine de ceux s'exécutant sur l'autre.

Plusieurs formalismes sont apparus pour raisonner sur de tels systèmes répartis. En particulier les ambients mobiles [CG00] de Luca Cardelli et Andrew Gordon se concentrent sur la notion de *localité*, les réductions de ces entités étant des réagencements de l'emboîtement de telles localités. Plus récemment, Robin Milner a proposé les bigraphes [JM04] comme formalisme permettant d'unifier les ambients mobiles et le π-calcul. La présente thèse repose quant à elle sur le Dπ-calcul, introduit par Matthew Hennessy et James Riely dans [HR02]. Ce calcul se distingue des autres en se posant comme une extension simple du π-calcul dans laquelle tous les processus sont placés dans une localité, sans que les localités puissent être emboîtées. Il est ainsi beaucoup plus facile de modéliser des comportements dans ce formalisme que dans les ambients mobiles, par exemple. Le choix d'une extension minimale du π-calcul permet d'hériter de la riche littérature existante et de mettre l'accent sur les nouvelles questions portant sur la répartition, telles que les droits d'accès.

Le Dπ-calcul étend le π-calcul en indiquant la *localité* dans laquelle un processus s'exécute de la façon suivante :

$$l[\![P]\!]$$

qui représente le processus P placé dans la localité l. La sémantique du langage ne portera donc plus sur des processus, comme P, mais sur des *systèmes*, comme $l[\![P]\!]$ pour prendre un exemple minimal, où tous les processus sont localisés. Naturellement, plusieurs processus peuvent être placés dans une même localité, identifiée par son nom et les systèmes peuvent se composer : dans le système

$$l[\![P_1]\!] \mid k[\![P_2]\!] \mid l[\![P_3]\!]$$

les deux processus P_1 et P_3 s'exécutent dans la même localité l tandis que le processus P_2 est situé dans la localité k, distincte de l. Par similitude avec CCS et le π-calcul, les trois processus P_1, P_2 et P_3 peuvent se réduire indépendamment les uns des autres. Mais la localisation joue un rôle majeur lorsque deux processus veulent communiquer : en effet, les communications ne peuvent prendre place qu'entre deux processus « voisins », situés dans la même localité. De sorte que la communication est régie par la réduction

$$l[\![a\,!\,\langle V\rangle\, P_1]\!] \mid l[\![a\,?\,(X : \mathsf{T})\, P_2]\!] \longrightarrow l[\![P_1]\!] \mid l[\![P_2\{^V/_X\}]\!]$$

L'autre ajout majeur par rapport au π-calcul est la possibilité pour les processus du Dπ-calcul de migrer entre localités. La règle suivante, d'une grande simplicité :

$$l[\![\mathsf{goto}\, k.\ P]\!] \longrightarrow k[\![P]\!]$$

codifie cette réduction.

Fidèle descendant de CCS et du π-calcul, le Dπ-calcul hérite des notions de congruence barbue et de bisimulation, à une petite transposition près. Le Dπ-calcul, sa sémantique par réductions et sa congruence barbue seront exposés en détails au chapitre 1. Mais on développera ces notions d'équivalence surtout dans un cadre typé (aux chapitres 2 et 4), où les capacités de discrimination de l'observateur sont décrites avec une grande précision.

Typage et contrôle de la mobilité

Le typage est une technique très répandue dans les langages de programmation (voir par exemple [Pie02]) essentiellement pour détecter des erreurs. Dans ce domaine, l'objectif que le typage tente d'atteindre peut se résumer par le slogan :

« Les programmes bien typés ne peuvent mal tourner. »

Le typage classe pour cela les valeurs du langage en types. Pour prendre un exemple élaboré et largement étudié, le typage des langages ML [DM82] garantit qu'une paire ne sera pas utilisée en lieu et place d'un triplet, une fonction attendant un entier en argument au lieu d'une fonction attendant un booléen, etc. Dans le cadre du Dπ-calcul, ces vérifications de bonne formation des systèmes consistent simplement à distinguer les canaux des localités, ainsi que les arités des canaux, elles ne nécessiteraient donc pas plus qu'un « sortage ».

Les typages que l'on met en place dans la présente thèse révèlent des propriétés nettement plus complexes qu'une simple bonne formation. Dans [PS96], Benjamin

Pierce et Davide Sangiorgi ont introduit un typage du π-calcul qui distingue les *capacités* d'émission et de réception sur un canal. Cette distinction autorise un processus à contrôler finement les droits qu'il accorde à ses interlocuteurs lors d'une communication. L'exemple de contrôle fourni dans [PS96] décrit un serveur d'impression qui attend des requêtes sur un canal *document*, donc de la forme suivante :

$$document\,?\,(d)\ \text{traitement de la requête...}$$

Si aucun contrôle spécifique n'est exercé sur l'usage du canal *document*, tout processus ayant le droit d'imprimer doit connaître le nom *document* et peut, par conséquent, l'utiliser pour détourner une requête en écoutant lui aussi sur ce canal. La séparation des capacités d'émission et de réception permet de contourner ce problème en ne rendant publique que la capacité d'émission sur *document*. Dans ces circonstances, le typage permet de démontrer que les documents ne seront récupérés que par l'imprimante.

Cette distinction des capacités est tout aussi pertinente dans le cadre du Dπ-calcul, pour décrire l'accès à un canal au sein d'une localité. Les capacités sont par conséquent conservées dans les typages mis en place dans des travaux comme [HR02] ou [HMR03]. La répartition complexifie cependant notablement le typage. Puisque les communications sont locales, il n'existe aucun lien entre les canaux utilisés dans deux localités distinctes. Les canaux sont ainsi naturellement liés à une localité. Le typage des canaux ne se contentera donc pas de séparer les différentes capacités disponibles mais mentionnera aussi la localité à laquelle le canal est attaché. Contrairement aux typages cités ci-dessus, on formalisera ce lien par des *sommes dépendantes*. Les sommes dépendantes permettent de spécifier, par exemple, que les deux canaux attendus par un processus doivent être disponibles dans une seule et même localité.

Par ailleurs, le formalisme du Dπ-calcul se fixe pour objectif la description et l'étude des comportements de processus pouvant migrer au sein d'un réseau de localités. Les typages existants rejettent cependant de nombreux *comportements récursifs*, par exemple lorsqu'un processus explore successivement une liste de localités en récupérant dans chacune le nom de la suivante. La difficulté se situe dans cette récupération dans une localité des ressources disponibles dans la suivante. On propose dans cette thèse un typage comportant des *types récursifs* pour autoriser ces comportements.

Exactement comme dans le cadre du π-calcul étudié dans [PS96], la distinction entre les capacités des canaux engendre un *sous-typage*. Le sous-typage est une relation d'ordre sur les types correspondant à l'inclusion des ensembles de valeurs. Dans le cas auquel on s'intéresse ici, un canal pour lequel les deux capacités d'émission et de réception sont disponibles est en particulier un canal sur lequel l'émission est autorisée. La théorie des types qui résulte du sous-typage en présence de types récursifs est relativement complexe. Dans [AC93], Roberto Amadio et Luca Cardelli ont développé une telle théorie pour le λ-calcul en utilisant des approximations finies des développements infinis des types récursifs. Cette théorie a été « digérée » et essentiellement réexprimée par Vladimir Gapeyev, Michael Levin et Benjamin Pierce dans [GLP03] en utilisant des types *coinductifs*.

L'approche coinductive est également poursuivie dans la présente thèse, car elle simplifie le développement de la théorie des types. En effet, cette théorie

ne s'arrête pas au sous-typage dans le cadre du Dπ-calcul : on utilise en effet extensivement les bornes inférieures d'ensembles de types. Ce besoin apparaît lorsqu'un processus reçoit par des voies différentes des capacités complémentaires : s'il reçoit au cours d'une communication la capacité d'émission sur un canal pour lequel il disposait déjà de la capacité de réception, on veut l'autoriser à utiliser les deux capacités simultanément. Cette nouvelle théorie des types, combinant tous les aspects des types évoqués ci-dessus, sera détaillée au chapitre 2.

La distinction des capacités nécessaires pour interagir avec un système et des capacités transmises au cours des communications conduit par ailleurs à une notion d'équivalence observationnelle plus précise. Les équivalences typées permettent de décrire finement les droits accordés à l'observateur. Sauf à « tricher », s'il n'a pas le droit de recevoir de message sur le canal c, il ne peut pas distinguer $l[\![c!\langle\rangle]\!]$ de $l[\![\mathsf{stop}]\!]$, puisqu'il est incapable d'interagir avec l'un comme avec l'autre. Il est donc nécessaire de prendre soigneusement en compte les capacités dont l'observateur dispose, mais aussi celles qu'il peut acquérir. On définira donc aussi au chapitre 2 une congruence barbue typée et une bisimilarité typée.

En poursuivant l'idée que le typage peut spécifier les droits transmis au cours d'une communication, on s'intéresse aussi dans cette thèse à une extension du Dπ-calcul et de son typage dans laquelle les migrations de processus doivent être dûment autorisées. En effet, selon les intuitions de programmation sous-jacentes, les processus sont des programmes qui peuvent migrer de machine en machine. Assez naturellement, on voudrait qu'une machine puisse contrôler quels programmes elle accepte, suivant une multitude de critères possibles, comme l'approche « bac à sable » de la machine virtuelle Java [Oak01], où de nombreuses fonctionnalités sont prohibées, ou encore l'approche « code porteur de preuve » [Nec97] de George Necula où le programme apporte un certificat de conformité à la politique de sécurité. Des extensions ont déjà été envisagées pour traiter le problème dans le Dπ-calcul, notamment dans [HMR03] et [HRY05]. Dans le premier, qui a fortement influencé l'extension que l'on propose ici, une nouvelle capacité associée aux localités est rajoutée au calcul pour spécifier si les migrations de processus vers cette localité sont autorisées. Le cadre présenté dans le second de ces travaux est nettement plus restrictif car il impose une limite stricte sur toutes les actions qu'un processus pourra effectuer après sa migration. La technique utilisée pour obtenir ce résultat est très compliquée et repose sur des communications d'ordre supérieur. On propose dans la présente thèse une première étape sur une voie médiane. L'idée que l'on explore est d'exiger d'un processus qui essaye de migrer d'une localité k vers une localité l qu'il possède un *passeport* correspondant précisément à cette migration $k \mapsto l$. Ces passeports représentent donc concrètement la confiance que porte la localité l à la localité k. Cette confiance permet de modéliser élégamment le comportement attendu au sein d'un réseau, par exemple, suivant des intuitions de confiance proches de celles exposées dans [RH03]. À l'image de [HMR03], on propose aussi une variante de ces passeports où l'origine d'un processus est ignorée, pour représenter le comportement de serveurs publics, par exemple. La théorie du calcul ainsi pourvu sera entièrement décrite au chapitre 4. L'essentiel de cette théorie porte sur les équivalences typées. L'ajout de passeports dans le calcul modifie en effet considérablement les possibilités d'interactions : un système peut se protéger des intrusions en délivrant soigneusement ses passeports. Les équivalences typées qui reposent sur les droits d'un observateur pour différencier les processus sont

automatiquement modifiées. On développera en totalité la théorie de comparaison des deux équivalences évoquées précédemment, renommées congruence barbue loyale et la bisimilarité loyale, dans ce cadre contrôlé.

On voudrait pouvoir aussi restreindre l'ensemble des ressources accessibles à un processus après la migration en fonction de l'identification que le passeport a permis d'effectuer. Cette étude ne sera évoquée qu'au titre de perspective du présent travail.

Plan de la thèse

Le premier chapitre présentera le Dπ-calcul, tel qu'hérité des travaux de Matthew Hennessy et James Riely, en s'intéressant aux différentes primitives qu'il comporte et à leur signification. On poursuivra cette présentation avec la syntaxe formelle des systèmes et la description des différentes notions nécessaires pour définir la sémantique par réductions du calcul. Enfin, on définira une équivalence observationnelle, à savoir la congruence barbue fermée par réduction en justifiant les différents points de sa définition.

Le chapitre 2 introduira un nouveau typage pour le Dπ-calcul. Après une série d'exemples illustrant des comportements que le typage permettra d'éviter, on décrira la syntaxe formelle de pré-types récursifs, comportant notamment les sommes dépendantes. Afin de décrire sous quelles conditions ces pré-types sont effectivement des types, on définira le pré-type coinductif correspondant à un pré-type récursif, puis le sous-typage sur les pré-types coinductifs. Le sous-typage sera la condition essentielle permettant de caractériser les types parmi les pré-types. Puis on développera la théorie complète des types coinductifs, c'est-à-dire la mise en place d'un critère, la compatibilité, identifiant les cas où deux types possèdent une borne inférieure, ainsi que l'extension de ce critère à des ensembles de types. On verra ensuite que cette théorie se transpose facilement au cas des types récursifs. Armés de ces notions, on pourra alors considérer le typage des systèmes, c'est-à-dire la vérification qu'un système utilise localités et canaux en respectant les types qui leur sont associés. On exposera ensuite quelques propriétés de ce typage, essentiellement le fait qu'un système bien typé le reste au cours de ses réductions. Puis on mettra en place une version typée de la congruence barbue, où un environnement de typage caractérise la connaissance de l'observateur sur les systèmes en cours de comparaison. Enfin, on expliquera comment définir une bisimilarité typée et on esquissera la preuve de sa coïncidence avec la congruence barbue typée.

Le chapitre 3 portera sur les liens entre la réplication et la récursion, les deux formes de comportements infinis, dans le cadre d'un calcul réparti. Après avoir décrit assez informellement comment remplacer l'une par l'autre, on donnera une implémentation formelle de la récursion utilisant la réplication et on expliquera pourquoi cette traduction n'est pas anodine dans un modèle réparti. On s'intéressera ensuite à la preuve que cette traduction transforme un système en un équivalent, au sens de la congruence barbue typée présentée au chapitre précédent. On mettra pour cela en place une technique de preuve appelée bisimulation modulo β et bisimulation forte. On exposera aussi une traduction plus complexe permettant de relier un système et sa traduction lorsqu'ils ont tous deux effectués une suite de réductions, et on terminera en prouvant que cette traduction est incluse dans la bisimilarité typée.

Enfin, le chapitre 4 se penchera sur le contrôle de la mobilité des processus dans le Dπ-calcul par l'usage de passeports. Après avoir esquissé un modèle où ces passeports permettent de contrôler très finement les droits accessibles aux processus après leur migration, on se restreindra à une partie simple, celle où la seule vérification effectuée porte sur l'origine des processus. On exposera donc l'impact de cet ajout sur la syntaxe des termes, leurs réductions ainsi que les types supplémentaires correspondant à ces passeports. On verra ensuite les modifications mineures que cela impose au typage des systèmes, et on prouvera, cette fois-ci en détails, plusieurs propriétés du typage, identiques ou similaires à celles du chapitre 2. Puis on définira une équivalence, descendante de la congruence barbue typée mais nettement plus complexe afin de tenir compte de l'impact des passeports sur les comportements raisonnablement observables d'un système. On proposera alors une définition de bisimilarité loyale respectant les passeports, largement plus compliquée, elle aussi, que sa variante du chapitre 2. On prouvera de façon extensive que la bisimilarité loyale coïncide avec la congruence loyale sur leur domaine commun, en vérifiant que la bisimilarité respecte les différentes conditions qui définissent la congruence, et, réciproquement, en utilisant ces conditions de la congruence pour détecter et simuler les actions sur lesquelles repose la bisimilarité. On finira en étudiant un peu plus précisément l'impact qu'auraient des passeports plus complexes sur le calcul.

Chapitre 1

Dπ

1.1 Présentation informelle du Dπ-calcul

Le Dπ-calcul [HR02], ou simplement Dπ, est un *calcul*[1] *de processus* introduit par Matthew Hennessy et James Riely pour étudier la *répartition* du calcul[2].

Un calcul est modélisé en Dπ par des *processus*. Afin de représenter la répartition du calcul, Dπ repose sur une notion de *localités*, ou de *sites*, de sorte que tous les processus sont annotés par la localité dans laquelle ils s'exécutent :

$$l[\![P]\!]$$

désigne le processus P placé dans la localité l.

Ces processus localisés peuvent être *composés en parallèle* pour former un *système* :

$$l[\![P]\!] \mid k[\![Q]\!]$$

comporte un processus P s'exécutant dans la localité l et un processus Q s'exécutant indépendamment dans la localité k.

À chaque instant, un des processus d'un système peut réaliser une étape de calcul interne ou bien interagir : une interaction met en jeu deux processus qui décident de se synchroniser et d'échanger un message. Pour avoir lieu, cette *communication* suppose qu'il existe un *canal* sur lequel un des deux processus émet un message tandis que l'autre processus écoute. Puisque la synchronisation entre deux processus est une opération complexe, Dπ impose que les communications soient locales, c'est-à-dire que les canaux sur lesquels elles s'effectuent soient eux aussi localisés et ne soient accessibles qu'aux processus s'exécutant dans leur localité. Pour prendre un exemple, le processus

$$c\,!\,\langle m\rangle\,\mathsf{stop}$$

qui procède à l'émission du message m sur le canal c puis s'arrête, et le processus « inverse »

$$c\,?\,(x)\,\mathsf{stop}$$

qui procède à la réception d'un message sur le même canal, peuvent être composés de la façon suivante :

$$l[\![c\,!\,\langle m\rangle\,\mathsf{stop}]\!] \mid l[\![c\,?\,(x)\,\mathsf{stop}]\!]$$

[1] au sens de formalisme mathématique sur lequel raisonner, voir note 3 page 7

[2] au sens de l'activité effectuée par une machine numérique, voir note 3 page 7

Ces deux processus étant placés dans la même localité, ils peuvent se synchroniser et échanger le message m sur le canal c. Le système qu'ils forment peut alors se *réduire* ainsi :

$$l[\![c\,!\,\langle m\rangle\,\mathsf{stop}]\!] \mid l[\![c\,?\,(x)\,\mathsf{stop}]\!] \longrightarrow l[\![\mathsf{stop}]\!] \mid l[\![\mathsf{stop}]\!]$$

Cet exemple illustre uniquement le mécanisme de synchronisation, puisque les deux processus s'arrêtent dès que le message est échangé. Mais on peut bien entendu aussi décrire des processus capables de poursuivre leur calcul après une communication. Ainsi, $c\,!\,\langle m\rangle$ peut *préfixer* n'importe quel processus P. Le processus $c\,!\,\langle m\rangle\,P$ commence son calcul en émettant le message m avant de déclencher sa *continuation* P, quels que soient les calculs que P effectue. Symétriquement $c\,?\,(x)\,Q$ attend un message sur le canal c pour déclencher sa continuation Q. Mais, pour qu'il y ait bien communication et pas seulement synchronisation, le processus récepteur Q doit apprendre le contenu du message. Le préfixe $c\,?\,(x)$ *lie* la *variable* x qui désignera dans le processus Q le message reçu au cours de cette communication, exactement comme les paramètres d'une fonction dans un langage de programmation. Lorsque la fonction est appelée, ses arguments formels sont remplacés par les arguments réels, ce que la réduction suivante représente dans le cadre du Dπ :

$$l[\![c\,!\,\langle m\rangle\,P]\!] \mid l[\![c\,?\,(x)\,Q]\!] \longrightarrow l[\![P]\!] \mid l[\![Q\{^{m}/_{x}\}]\!]$$

où $Q\{^{m}/_{x}\}$ est le processus Q dans lequel les x sont substitués par m. Au même titre que les paramètres d'une fonction, les x concernés par cette substitution ne sont que les *occurrences libres* de x dans Q. On dit alors que la *portée* de x est Q. Puisque le processus $c\,?\,(x)\,Q$ attend la réception d'un message avant d'exécuter Q, la variable x aura été substituée avant que Q ne cherche à l'« utiliser ».

L'indépendance des processus entre eux implique que cette communication pourra avoir lieu quels que soient les autres processus présents, par exemple :

$$l[\![c\,!\,\langle m\rangle\,P]\!] \mid k[\![R_1]\!] \mid l[\![c\,?\,(x)\,Q]\!] \mid l[\![R_2]\!] \longrightarrow l[\![P]\!] \mid k[\![R_1]\!] \mid l[\![Q\{^{m}/_{x}\}]\!] \mid l[\![R_2]\!]$$

Cette situation plus complexe illustre également que la proximité syntaxique des deux processus qui interagissent dans l'exemple précédent n'est pas indispensable.

Si ces deux primitives de communication étaient les seules opérations permises pour les processus, Dπ ne permettrait de décrire comme systèmes répartis que des systèmes dans lesquels les processus sont distribués une fois pour toute sur l'ensemble des localités existantes. Dπ fournit par conséquent d'autres primitives que la communication pour décrire le calcul.

Dπ introduit de la dynamique notamment en permettant aux processus de migrer d'une localité à l'autre. Puisque la localisation y prend la forme très simple d'une annotation indiquant le nom de la localité dans laquelle le processus s'exécute, l'opération de migration s'y exprime aisément :

$$l[\![\mathsf{goto}\ k.\ P]\!] \longrightarrow k[\![P]\!]$$

Un autre aspect dynamique de Dπ est la possibilité de générer de nouveaux noms, qu'il s'agisse de noms de localités ou de canaux. Les noms ainsi créés sont alors distincts de tout autre nom préexistant, on dit qu'ils sont *inédits*[3]. Ce

[3]Le terme consacré en anglais pour cette notion est *fresh*.

mécanisme est souvent utilisé en Dπ pour permettre d'établir une connexion privée entre deux entités. Par exemple, le système

$$l[\![c\,?\,(x)\,x\,!\,\langle secret\rangle]\!] \mid l[\![\mathsf{newchan}\ cp\ \mathsf{in}\ c\,!\,\langle cp\rangle\ cp\,?\,(ys)\ P]\!]$$

se réduit après un certain nombre d'étapes de calcul en

$$(\mathsf{new}\ cp)(l[\![cp\,!\,\langle secret\rangle]\!] \mid l[\![cp\,?\,(ys)\ P]\!])$$

puis, en une étape supplémentaire, en

$$(\mathsf{new}\ cp)(l[\![\mathsf{stop}]\!] \mid l[\![P\{^{secret}/_{ys}\}]\!])$$

c'est-à-dire que les deux processus ont pu communiquer en utilisant le canal privé *cp* pour échanger un *secret*.

La création de noms peut aussi permettre de modifier l'architecture du système. La localisation des processus permet ainsi d'isoler certaines parties d'un calcul, en générant de nouvelles localités dans lesquelles les effectuer. Prenons un exemple : un système qui veut effectuer un long calcul de façon totalement indépendante de son activité normale peut prendre la forme

$$l[\![\mathsf{newloc}\ temp, (résultat)\ \mathsf{with}\ \cdots \textsc{calcul} \cdots résultat\,!\,\langle r\rangle\ \mathsf{in}\ P]\!]$$

Ce système génère en effet une nouvelle localité *temp* et un nouveau canal *résultat* dans *temp*. Il déclenche ensuite dans cette nouvelle localité le processus $\cdots \textsc{calcul} \cdots résultat\,!\,\langle r\rangle$ qui effectue le calcul et rend disponible son résultat sur le canal *résultat*. Le processus P décrit quant à lui l'activité normale qui se poursuit dans la localité l. Le système se réduit en effet en

$$(\mathsf{new}\ temp)\ (\mathsf{new}\ résultat)\ temp[\![\cdots \textsc{calcul} \cdots résultat\,!\,\langle r\rangle]\!] \mid l[\![P]\!]$$

Dès que le résultat du calcul lancé dans la localité *temp* s'avère nécessaire, le processus P peut aller le chercher de la façon suivante :

$$\mathsf{goto}\ temp.\ résultat\,?\,(x)\ \mathsf{goto}\ l.\ P'$$

où P' donne le comportement de P une fois que le résultat du calcul est connu.

Dπ fournit bien entendu également des procédés pour décrire des comportements infinis, indispensables pour modéliser un serveur, par exemple. Deux primitives sont disponibles : la réplication et la récursion. La première, notée $*P$ pour le processus P répliqué, est particulièrement adaptée à la description d'un serveur car elle se réduit de la façon suivante :

$$l[\![*P]\!] \longrightarrow l[\![P]\!] \mid l[\![*P]\!]$$

Cette construction permet donc de modéliser très naturellement un serveur qui gérerait chaque requête par un nouveau processus, tous ces processus étant identiques. Illustrons ce mécanisme par un serveur rudimentaire qui génère un nouveau nom de canal pour chaque requête :

$$l[\![*req\,?\,(x)\ \mathsf{newchan}\ c\ \mathsf{in}\ x\,!\,\langle c\rangle]\!]$$

qui peut donc se réduire en

$$l[\![*req\,?\,(x)\ \mathsf{newchan}\ c\ \mathsf{in}\ x\,!\,\langle c\rangle]\!] \mid l[\![req\,?\,(x)\ \mathsf{newchan}\ c\ \mathsf{in}\ x\,!\,\langle c\rangle]\!]$$

En composant donc ce serveur avec le client $l[\![req\,!\,\langle r\rangle\, r\,?\,(x)\, Q]\!]$ on obtient le système

$$\begin{array}{ll} & l[\![*req\,?\,(x)\,\mathsf{newchan}\,c\,\mathsf{in}\,x\,!\,\langle c\rangle]\!] \\ \mid & l[\![req\,?\,(x)\,\mathsf{newchan}\,c\,\mathsf{in}\,x\,!\,\langle c\rangle]\!] \\ \mid & l[\![req\,!\,\langle r\rangle\, r\,?\,(x)\, Q]\!] \end{array}$$

où le processus $req\,?\,(x)\,\mathsf{newchan}\,c\,\mathsf{in}\,x\,!\,\langle c\rangle$ peut servir le client $req\,!\,\langle r\rangle\, r\,?\,(x)\, Q$, tandis que le serveur est toujours apte à traiter d'autres requêtes en parallèle puisque $l[\![*req\,?\,(x)\,\mathsf{newchan}\,c\,\mathsf{in}\,x\,!\,\langle c\rangle]\!]$ est conservé intact.

La récursion[4] est une approche plus complexe des comportements infinis. Le processus

$$(\mathsf{rec}\ Z\,(X)\,.\,P)\,\langle V\rangle$$

effectue en effet deux opérations simultanément : il déclare le processus récursif proprement dit $\mathsf{rec}\ Z\,(X)\,.\,P$ et invoque une première instance de ce processus avec V comme argument. Cela signifie que le paramètre X sera substitué par V dans P avant que P ne commence son exécution. Le processus $\mathsf{rec}\ Z\,(X)\,.\,P$ est récursif au sens où P peut contenir des appels récursifs de la forme

$$Z\,\langle V'\rangle$$

où V' est l'argument de l'appel.

Le Dπ-calcul dispose enfin de trois dernières primitives. La première que l'on n'ait pas encore mentionné bien qu'elle soit fort naturelle, est simplement la composition parallèle des processus. Au même titre que l'on peut composer deux processus localisés $l[\![P]\!]$ et $k[\![Q]\!]$ pour former le système $l[\![P]\!] \mid k[\![Q]\!]$, on peut composer directement deux processus P et Q pour obtenir un processus $P \mid Q$. On utilise la même notation pour les deux compositions dans la mesure où elles se correspondent très intuitivement et sans ambiguïté. En effet, la première étape de calcul du système $l[\![P \mid Q]\!]$ sépare les deux processus pour donner $l[\![P]\!] \mid l[\![Q]\!]$.

La deuxième primitive encore inconnue, $\mathsf{here}\,(x)$, apporte la possibilité de détecter la localité dans laquelle un processus s'exécute, en liant le nom de cette localité à la variable x dans la continuation, suivant la réduction :

$$l[\![\mathsf{here}\,(x)\,P]\!] \longrightarrow l[\![P\{^{l}/_{x}\}]\!]$$

Enfin, la dernière primitive est traditionnelle en programmation : le test. Dans un souci de simplicité, on restreindra les valeurs que les processus manipulent aux objets fondamentaux du Dπ, à savoir les localités et les canaux[5]. Par conséquent, les seuls tests possibles dans le langage portent sur l'égalité ou l'inégalité de ces valeurs. Par exemple, quelle que soit la localité dans laquelle il s'exécute, le processus $\mathsf{if}\ a_1 = a_2\ \mathsf{then}\ P\ \mathsf{else}\ Q$ se réduit vers P si a_1 et a_2 sont égaux et vers Q dans le cas contraire.

Exemple complet

Afin d'illustrer le fonctionnement d'une grande partie des primitives du Dπ-calcul, décrivons un processus prospecteur qui parcourrait un réseau en quête d'*or* !

Le prospecteur explore le réseau de la façon suivante :

[4] Les liens existants entre la réplication et la récursion seront explorés en détail au chapitre 3.

[5] Le chapitre 4 introduira de nouveaux objets, les passeports, sur lequel les tests seront bien entendu également possibles.

1. il regarde quelle valeur est disponible dans la localité où il se trouve ;
2. si cette valeur est l'*or* espéré, sa quête s'achève là : il peut retourner à son *campement* et y rapporter le nom de la localité où il a trouvé de l'*or* ;
3. sinon il poursuit son exploration du réseau : il récupère les coordonnées d'une localité voisine, y migre et reprend son investigation à la première étape.

Ce processus sera donc naturellement récursif. En insistant sur l'indentation pour en clarifier la lecture, ce processus pourrait donc prendre la forme suivante :

$$
\begin{array}{l}
l\ [\![\\
\quad (\\
\quad\quad \mathsf{rec}\ Z\,(x_{\mathrm{val}}, x_{\mathit{vois}})\,. \\
\quad\quad\quad x_{\mathrm{val}}\,?\,(x) \\
\quad\quad\quad \mathsf{if}\ x = \mathit{or} \\
\quad\quad\quad \mathsf{then\ here}\,(y)\ \mathsf{goto}\ \mathit{campement}.\ \mathit{locOr}\,!\,\langle y\rangle \\
\quad\quad\quad \mathsf{else}\ x_{\mathit{vois}}\,?\,(y, X)\ \mathsf{goto}\ y.\ Z\,\langle X\rangle \\
\quad)\,\langle \mathit{val}, \mathit{vois}\rangle \\
]\!]
\end{array}
$$

Ce processus récursif commence par instancier ses arguments formels x_{val} et x_{vois} et déplier l'appel récursif :

$$
\begin{array}{l}
\longrightarrow\ l\ [\![\\
\quad\quad \mathit{val}\,?\,(x) \\
\quad\quad \mathsf{if}\ x = \mathit{or} \\
\quad\quad \mathsf{then\ here}\,(y)\ \mathsf{goto}\ \mathit{campement}.\ \mathit{locOr}\,!\,\langle y\rangle \\
\quad\quad \mathsf{else}\ \mathit{vois}\,?\,(y, X)\ \mathsf{goto}\ y.\ (\mathsf{rec}\ Z\,(x_{\mathrm{val}}, x_{\mathit{vois}})\,.\ \cdots)\,\langle X\rangle \\
\quad]\!]
\end{array}
$$

Il récupère alors la valeur disponible sur le canal local *val* :

$$
\begin{array}{l}
\longrightarrow\ l\ [\![\\
\quad\quad \mathsf{if}\ \mathit{valeur\_disponible} = \mathit{or} \\
\quad\quad \mathsf{then\ here}\,(y)\ \mathsf{goto}\ \mathit{campement}.\ \mathit{locOr}\,!\,\langle y\rangle \\
\quad\quad \mathsf{else}\ \mathit{vois}\,?\,(y, X)\ \mathsf{goto}\ y.\ (\mathsf{rec}\ Z\,(x_{\mathrm{val}}, x_{\mathit{vois}})\,.\ \cdots)\,\langle X\rangle \\
\quad]\!]
\end{array}
$$

Si cette valeur est celle recherchée, *or*, il rentre à son *campement* et rapporte sa trouvaille :

$$
\begin{array}{ll}
\longrightarrow & l[\![\mathsf{here}\,(y)\ \mathsf{goto}\ \mathit{campement}.\ \mathit{locOr}\,!\,\langle y\rangle]\!] \\
\longrightarrow & l[\![\mathsf{goto}\ \mathit{campement}.\ \mathit{locOr}\,!\,\langle l\rangle]\!] \\
\longrightarrow & \mathit{campement}[\![\mathit{locOr}\,!\,\langle l\rangle]\!]
\end{array}
$$

Si la valeur disponible n'est pas celle espérée, il récupère le nom de la localité voisine k ainsi que les noms des canaux correspondants à *val* et *vois* chez cette voisine :

$$
\begin{array}{ll}
\longrightarrow & l[\![\mathit{vois}\,?\,(y, X)\ \mathsf{goto}\ y.\ (\mathsf{rec}\ Z\,(x_{\mathrm{val}}, x_{\mathit{vois}})\,.\ \cdots)\,\langle X\rangle]\!] \\
\longrightarrow & l[\![\mathsf{goto}\ k.\ (\mathsf{rec}\ Z\,(x_{\mathrm{val}}, x_{\mathit{vois}})\,.\ \cdots)\,\langle \mathit{val}_k, \mathit{vois}_k\rangle]\!]
\end{array}
$$

Fig. 1.1 Identifiants, valeurs et motifs

$u, v ::=$	*Identifiants*
$a, b, c, d, k, l, \ldots$	Noms
$x, y, \ldots$	Variables
$s, t ::=$	*Identifiants étendus*
$u, v, \ldots$	Identifiants
$Z, \ldots$	Variables de récursion
$U, V ::=$	*Valeurs*
u	Identifiants
$(V_1, \ldots, V_n)$	n-uplets
$S, T ::=$	*Valeurs étendues*
s	Identifiants étendus
$(S_1, \ldots, S_n)$	n-uplets
$X ::=$	*Motifs*
$x, y, \ldots$	Variables
$(X_1, \ldots, X_n)$	n-uplets où chaque variable apparaît au plus une fois

Il finit alors en migrant chez la voisine pour y déclencher une nouvelle instance de sa récursion :

$$\longrightarrow k[\![(\mathsf{rec}\ Z\,(x_{val}, x_{vois})\,.\ \cdots)\,\langle val_k, vois_k\rangle]\!]$$

Partant des intuitions présentées dans cette section, définissons formellement le Dπ-calcul.

1.2 Syntaxe et sémantique par réductions du Dπ-calcul

On fonde le Dπ-calcul sur un ensemble infini de noms, qu'il s'agisse de canaux ou de localités, que l'on notera dans la suite a, b, etc. On utilisera préférentiellement c, d, etc. quand on voudra insister sur le fait que ces noms correspondent à des canaux et k, l, etc. pour des localités. Comme on l'a indiqué dans la description informelle du calcul à propos des préfixes de réception $a\,?\,(X)$, on considérera aussi un jeu, également infini, de variables, que l'on notera x, y, etc. On autorisera indistinctement dans la syntaxe formelle des systèmes l'utilisation de variables ou de noms pour décrire les processus. Et puisque la majorité des primitives du langage peuvent mentionner indifféremment noms et variables, on utilisera u, v, etc. dans la suite pour désigner un *identifiant*, qu'il s'agisse d'un nom ou d'une variable. De plus, il existe une autre famille de variables dans le langage, déjà utilisée pour décrire le prospecteur : on parlera de *variable de récursion* pour les variables apparaissant dans les définitions des processus récursifs. On les notera souvent Z. La syntaxe de ces entités est résumée en figure 1.1. On considérera un calcul privé de valeurs de « base » comme des entiers, des booléens, etc. par souci de simplicité.

Fig. 1.2 Systèmes et processus

$M ::=$	*Systèmes*
$l[\![P]\!]$	Processus localisé
$M_1 \mid M_2$	Composition parallèle
$(\mathsf{new}\, a : \mathsf{E})\, M$	Portée de nom
$\mathbf{0}$	Système inactif
$P ::=$	*Processus*
$u\,!\,\langle V\rangle\, P$	Écriture
$u\,?\,(X : \mathsf{T})\, P$	Lecture
$\mathsf{goto}\, u.\, P$	Migration
$\mathsf{if}\, u_1 = u_2\, \mathsf{then}\, P_1\, \mathsf{else}\, P_2$	Comparaison
$\mathsf{newchan}\, c : \mathsf{C}\, \mathsf{in}\, P$	Création de canal
$\mathsf{newloc}\, k, (c_1, \ldots, c_n) : \mathsf{L}\, \mathsf{with}\, P_k\, \mathsf{in}\, P$	Création de localité
$P_1 \mid P_2$	Composition parallèle
$\mathsf{here}\,(x)\, P$	OùSuisJe
$*P$	Réplication
$(\mathsf{rec}\, Z : \mathsf{R}\,(X)\,.\, P)\,\langle V\rangle$	Récursion
$Z\,\langle V\rangle$	Appel récursif
stop	Terminaison

Ces familles syntaxiques permettent de définir les *processus*, notés P, Q, etc. dont la syntaxe formelle est donnée figure 1.2, ces processus devant être placés dans une localité pour former des *systèmes*, décrits dans la même figure. Lorsque deux processus P et Q sont annotés par un même nom de localité, par exemple dans le système $l[\![P]\!] \mid l[\![Q]\!]$, ils sont effectivement situés dans la même localité. La syntaxe formalise aussi le fait que les localités ne peuvent être emboîtées, la structure spatiale qui naît de la répartition est donc plate : toutes les localités sont au même niveau.

Les idées correspondant aux primitives que peuvent comporter les processus sont déjà apparues dans la présentation informelle du calcul. Comme le laisse deviner la réception $vois\,?\,(y, X)$ de l'exemple du prospecteur, au lieu de n'autoriser que le préfixe d'émission $u\,!\,\langle v\rangle$ qui correspond à l'échange d'une donnée simple c'est-à-dire un seul nom, on permet la transmission de *valeurs* plus complexes, comme les n-uplets. On notera souvent en abrégé $\vec{u}$ un n-uplet de la forme $(u_1, \ldots, u_n)$ et $\tilde{u}$ l'ensemble des identifiants qui le composent $\{u_1, \ldots, u_n\}$. On autorise aussi l'emboîtement de ces n-uplets de sorte que la transmission de valeurs arborescentes soit facile. On écrit par conséquent le préfixe d'émission sous la forme $u\,!\,\langle V\rangle$ où l'on dénote par U, V, etc. les valeurs du langage. Symétriquement, la réception de ces valeurs doit permettre de les décomposer et donc de lier de simples variables aux valeurs qui sont émises. On rajoute donc pour cela la catégorie syntaxique des *motifs*, qui reprend la structure des valeurs en ne contenant que des variables et en assurant que celles-ci n'apparaissent qu'une fois dans le motif, les questions que poseraient des motifs plus puissants étant hors du propos du présent travail. En dénotant les motifs par X, Y, etc. le préfixe de réception devient donc $u\,?\,(X : \mathsf{T})$. On y mentionne en effet dès à présent l'information de type T, qui est le type associé au motif X.

On mentionne également les types pour les autres préfixes liants :

- $\mathsf{newchan}\, c : \mathsf{C}$ qui génère et lie un nom de canal (c), mentionne le type C ;
- $\mathsf{newloc}\, k, (c_1, \ldots, c_n) : \mathsf{L}$ qui génère et lie un nom de localité (k) et des noms de canaux[6] ($c_1, \ldots, c_n$), mentionne le type L ;
- $(\mathsf{rec}\ Z : \mathsf{R}(X).\ \cdot)\langle V \rangle$ qui définit un processus récursif et lie donc le paramètre X, mentionne le type R.

On utilise des lettres différentes pour désigner les types apparaissant dans ces constructions afin de souligner que ces types prendront des formes différentes qui seront décrites dans la suite. Ces annotations de types peuvent être simplement ignorées dans le cadre de ce chapitre mais elles prendront tout leur intérêt dans les chapitres suivants.

On veut maintenant donner une sémantique au calcul que l'on vient de définir. Cependant, il est nécessaire de formaliser quelques notions préalables avant de pouvoir donner les règles de réduction. En particulier, il faut établir formellement les notions de variable libre et de nom libre. Nous avons évoqué dans la présentation du calcul la liaison d'un nom ou d'une variable par un préfixe. Ainsi, une occurrence donnée d'un nom a est liée dans un processus ou un système si elle apparaît sous un des préfixes générateurs de noms ($\mathsf{new}\, a : \mathsf{E}$), $\mathsf{newchan}\, a : \mathsf{C}$, $\mathsf{newloc}\, a, (\vec{c}) : \mathsf{L}$ ou $\mathsf{newloc}\, k, (\ldots, a, \ldots) : \mathsf{L}$. On notera au passage que $\mathsf{newloc}\, k, \vec{c} : \mathsf{L}\ \mathsf{with}\ P_k\ \mathsf{in}\ P$ lie les noms k et $\tilde{c}$ à la fois dans les processus P_k et P : le fait de créer des noms de canaux en même temps que la localité est en effet intéressant parce qu'ils sont automatiquement connus dans P. Quant aux variables, une occurrence de x est liée dans un processus si elle apparaît sous un préfixe $u\,?\,(X)$ où x apparaît dans le motif X.

Plus que la notion d'occurrences liées, les notions de variables libres et de noms libres sont indispensables pour définir la sémantique du calcul. On dit qu'une occurrence d'un nom ou d'une variable est libre si elle n'est pas située sous un lieur, c'est-à-dire si elle n'est pas liée. Cette notion de liberté joue un rôle majeur dans le cadre du λ-calcul, elle est donc très bien connue. Elle s'adapte naturellement au Dπ où cohabitent deux espèces distinctes : les noms et les variables, chacune ayant ses lieurs. C'est pourquoi on ne définira pas ici les ensembles de noms libres ou de variables libres d'un système. Cependant, dans un souci d'exhaustivité, on donne en annexe la formalisation des ensembles de noms libres et de variables libres, définis naturellement par induction sur les processus et systèmes. Cette approche est parfaitement adaptée puisque la portée des lieurs est syntaxique.

Notation 1.1 (Noms et variables libres)**.** *On notera* $\mathsf{fn}(P)$ *et* $\mathsf{fv}(P)$ *respectivement les noms et variables* libres *du processus* P*. Ces ensembles sont aussi définis sur les systèmes et les valeurs. Ces ensembles sont définis formellement dans les figures A.1 (page 178) et A.2 (page 179).*

On notera $\mathsf{frv}(P)$ *l'ensemble des variables de récursion libres du processus* P*, construit de façon similaire*[7]*.*

On notera $\mathsf{fid}(P)$ *l'ensemble des identifiants étendus libres, c'est-à-dire l'ensemble des noms, variables et variables de récursion libres de* P*, soit* $\mathsf{fn}(P) \cup \mathsf{fv}(P) \cup \mathsf{frv}(P)$*.*

[6] Un exemple de la section 1.1 utilise la facilité d'écriture que permet la création de canaux en même temps qu'une localité. Cette fonctionnalité est cependant présente dans le langage parce qu'elle s'avérera indispensable au chapitre 4 où les migrations peuvent être interdites.

[7] Leur seule occurrence possible est l'appel récursif $Z\langle V \rangle$, et leur seul lieur $\mathsf{rec}\ Z$.

Bien que les notions de liberté et de liaison soient antinomiques, une variable ou un nom donné peut apparaître à la fois lié et libre à deux endroits distincts d'un processus ou d'un système. Par ailleurs, ces notions de liberté et de liaison jettent une petite lumière sur les différences qui existent entre variables et noms. Alors que les lieurs de noms les génèrent, les variables ne peuvent être liées que par le préfixe de réception. Les variables ont donc vocation à toujours être substituées par des noms au cours de l'exécution avant d'être vraiment « utilisées ». C'est là la différence essentielle entre variables et noms dans le calcul, et c'est pourquoi on se contentera de définir la sémantique des systèmes dans lesquels aucune variable n'est libre.

Les variables de récursion partagent avec les simples variables la caractéristique de ne pas avoir beaucoup de sens quand elles ne sont pas liées. On a en effet besoin de savoir quel processus déclencher quand on cherche à exécuter un appel récursif. Les variables de récursion libres seront donc prohibées.

Définition 1.2 (Processus et systèmes clos)**.** *Un système M est dit* clos *si* $\mathsf{fv}(M) \cup \mathsf{frv}(M)$ *est vide. Le même critère s'applique pour un processus.*

On supposera implicitement dans la suite tous les systèmes clos.

Si on peut contraindre les variables à être toujours liées c'est que la sémantique d'un processus $a\,?\,(x)\,P$ est définie en fonction de $P\{^{v}/_{x}\}$ c'est-à-dire le processus P dans lequel x a été substitué par v. De la même façon le processus $(\mathsf{rec}\ Z\,(V)\,.\ P)\,\langle X\rangle$ se réduit au processus $P\{^{V}/_{X}\}[^{\mathsf{rec}\ Z(X).\ P}/_{Z}]$, c'est-à-dire le processus dans lequel le paramètre X a été instancié par V et la variable de récursion Z substituée par une nouvelle instance du processus récursif, prête à se réduire à son tour.

Ces deux sortes de substitution, de variables simples par des valeurs et de variables de récursion par des processus complets, sont des substitutions *sans capture*, c'est-à-dire que les noms ou variables par lesquels on substitue ne peuvent pas se retrouver liés après substitution.

Pour appréhender ce problème de capture, considérons le système $M = l[\![a\,?\,(x : \mathsf{T})\,\mathsf{newchan}\,c : \mathsf{C}\,\mathsf{in}\,x\,!\,\langle\rangle]\!]$. Ce système attend un nom de canal dénoté par la variable x dans la suite du processus, crée un nouveau canal qu'il appelle c, et envoie un message sur le canal dénoté par x. Ce système peut communiquer avec un système $l[\![a\,!\,\langle c\rangle]\!]$ qui se contente de lui envoyer le nom du canal libre c. On veut donc pouvoir appliquer une substitution au processus $\mathsf{newchan}\,c : \mathsf{C}\,\mathsf{in}\,x\,!\,\langle\rangle$ pour remplacer l'occurrence de x par un c. Mais si on procédait à une simple substitution de toutes les occurrences de x par des c, on aboutirait au processus $\mathsf{newchan}\,c : \mathsf{C}\,\mathsf{in}\,c\,!\,\langle\rangle$. Le sens du processus s'en trouverait changé, puisque le système $l[\![\mathsf{newchan}\,c : \mathsf{C}\,\mathsf{in}\,c\,!\,\langle\rangle]\!]$ crée un nouveau canal, incidemment nommé c, et émet un message sur ce canal, et non plus sur le canal c dont la portée dépasse le processus lui-même. On dit que le nom c est capturé par le lieur $\mathsf{newchan}\,c : \mathsf{C}\,\mathsf{in}\ldots$ Cette « collision » entre les noms identiques de deux canaux distincts modifie donc gravement la signification du système.

C'est pourquoi la définition de la substitution doit prendre garde de ne passer sous les lieurs uniquement quand les noms et variables liés sont distincts des identifiants de la substitution. À cause de ce choix, la substitution n'est pas définie dans les cas où elle pourrait provoquer une collision, comme dans le cas $(\mathsf{newchan}\,c : \mathsf{C}\,\mathsf{in}\,x\,!\,\langle c\rangle)\{^{c}/_{x}\}$.

Il est cependant indispensable de donner un sens à cette substitution, sans quoi des processus pourraient bloquer à la réception de certains messages. Le

nom c apparaissant dans le lieur du processus précédent est par ailleurs arbitraire : la signification intuitive que nous avons donnée à ce processus dans la présentation du langage ne marque pas une différence nette entre lui et le processus $\mathsf{newchan}\, d : \mathsf{C}\, \mathsf{in}\, x\,!\,\langle d\rangle$. Il s'agit en effet toujours de générer un nouveau nom de canal et de le transmettre sur le canal x. Mais on voit que ce processus a l'avantage de permettre l'application de la substitution $\{^{c}/_{x}\}$. Le même type d'opération de renommage des noms est nécessaire pour les substitutions des appels récursifs, afin d'éviter de capturer les noms et variables du processus qui remplacent la variable de récursion Z.

Ces substitutions sans capture se définissent facilement par induction structurelle sur les processus auxquels elles sont appliquées. Exactement comme les définitions des noms et variables libres, la substitution sans capture des variables est héritée du λ-calcul où elle est très bien connue. Dans un souci d'exhaustivité, elle est donnée en annexe, dans la figure A.3 (page 180). La substitution de motifs par des valeurs est une simple extension de cette définition sur les variables et les identifiants. La définition formelle de la décomposition d'une substitution de valeurs en une succession de substitutions d'identifiants sera donnée dans la suite.

Quant à la substitution sans capture des appels récursifs, le principal cas intéressant est naturellement :

$$(Z\,\langle V\rangle)[^{\mathsf{rec}\ Z:\mathsf{R}(X).\ Q}/_{Z}] = (\mathsf{rec}\ Z : \mathsf{R}\,(X)\,.\,Q)\,\langle V\rangle$$

et le principal point est évidemment d'éviter les captures des noms et variables libres du processus récursif.

$$(\mathsf{newchan}\, c : \mathsf{C}\, \mathsf{in}\, P)[^{\mathsf{rec}\ Z:\mathsf{R}(X).\ Q}/_{Z}] = \mathsf{newchan}\, c : \mathsf{C}\, \mathsf{in}\, P[^{\mathsf{rec}\ Z:\mathsf{R}(X).\ Q}/_{Z}]$$

quand $c \notin \mathsf{fn}(Q)$. La figure A.4 (page 180) donnée en annexe précise quelques autres cas.

Le renommage des identifiants liés peut toujours être effectué puisque les ensembles de variables et de noms sont infinis. Il établit ainsi une équivalence sur les processus et systèmes, bien connue et habituellement appelée α-équivalence. Afin d'éviter un blocage d'un processus au niveau d'une substitution, nous identifierons dans la suite tous les systèmes et processus à α-équivalence près. Formellement, chaque système et chaque processus sera donc vu comme un représentant de sa classe d'α-équivalents. Nous utiliserons aussi une convention simplificatrice, habituellement appelée convention de Barendregt.

Convention 1.3 (Barendregt)**.** *Tous les noms et variables liés seront supposés distincts deux à deux et distincts de tous les noms et variables libres.*

Dans le cadre de cette convention, la substitution est toujours définie. En fait, effectuer une substitution devient une simple question de remplacement de toutes les occurrences de la variable ou du nom substitué puisqu'il n'y a plus de risques de captures des noms et variables. Bien entendu, après chaque substitution, il faut éventuellement choisir une nouvelle représentation du système substitué pour continuer à respecter la convention.

La sémantique du calcul repose également sur une autre notion d'équivalence, appelée *congruence structurelle*. Voyons où cette équivalence s'avère nécessaire pour donner un sens aux systèmes. Comme la présentation informelle de Dπ

Fig. 1.3 Règles de la congruence structurelle

$$M \mid \mathbf{0} \equiv M$$
$$M_1 \mid M_2 \equiv M_2 \mid M_1$$
$$M_1 \mid (M_2 \mid M_3) \equiv (M_1 \mid M_2) \mid M_3$$
$$M_1 \mid (\mathsf{new}\, a : \mathsf{E})\, M_2 \equiv (\mathsf{new}\, a : \mathsf{E})(M_1 \mid M_2) \quad \text{quand } a \notin \mathsf{fn}(M_1)$$
$$(\mathsf{new}\, a : \mathsf{E}_a)\,(\mathsf{new}\, b : \mathsf{E}_b)\, M \equiv (\mathsf{new}\, b : \mathsf{E}_b)\,(\mathsf{new}\, a : \mathsf{E}_a)\, M \quad \text{quand } a \neq b,\ a \notin \mathsf{fn}(\mathsf{E}_b),\ b \notin \mathsf{fn}(\mathsf{E}_a)$$

le laisse comprendre, les communications sont autorisées par une règle de la sémantique affirmant qu'un système M de la forme $l[\![a\,!\,\langle V\rangle\, P_1]\!] \mid l[\![a\,?\,(X : \mathsf{T})\, P_2]\!]$ peut toujours effectuer une communication sur le canal a dans la localité l pour donner le système $l[\![P_1]\!] \mid l[\![P_2\{^V/_X\}]\!]$. Cependant, on veut naturellement que le système $l[\![a\,!\,\langle V\rangle\, P_1]\!] \mid (\mathsf{new}\, b : \mathsf{E}_b)\, l[\![a\,?\,(X : \mathsf{T})\, P_2]\!]$, bien qu'il ne se présente pas exactement comme le système M, puisse lui aussi effectuer une communication. On définit pour cela une congruence qui met en relation des systèmes auxquels on veut associer une même sémantique. Pour ce faire, on introduit quelques notions préalables.

Définition 1.4 (Contexte)**.** *On appelle* contexte *un « système avec un trou », c'est-à-dire de la forme :*

$\mathfrak{C} ::=$	*Contextes*
$[\cdot]$	*« Trou »*
$\mathfrak{C} \mid M$	
$M \mid \mathfrak{C}$	
$(\mathsf{new}\, a : \mathsf{E})\, \mathfrak{C}$	

On note $\mathfrak{C}[M]$ le système constitué en remplaçant le trou $[\cdot]$ du contexte $\mathfrak{C}$ par le système M.

Notons que l'on ne prend pas les mêmes précautions dans la définition des contextes que la définition de la substitution : alors que la substitution évite les captures de noms (et de variables) en évitant de définir $(\mathsf{newchan}\, c : \mathsf{C}\, \mathsf{in}\, P)\{^c/_x\}$ par exemple, le contexte $(\mathsf{new}\, c : \mathsf{E})[\cdot]$ dans lequel on place le système $l[\![c\,!\,\langle\rangle]\!]$ forme $(\mathsf{new}\, c : \mathsf{E})\, l[\![c\,!\,\langle\rangle]\!]$ où c est lié. Cette différence est voulue : un système ne contrôle pas l'univers dans lequel il est placé, ses noms libres autorisant les interactions avec cet environnement. En revanche les choix arbitraires de noms et variables liés dans le système ne doivent pas entrer en conflit avec les substitutions lors de l'exécution du système.

La notion de contexte permet de formaliser la contextualité d'une relation.

Définition 1.5 (Relation contextuelle)**.** *Une relation $\mathcal{R}$ est* contextuelle *si, quels que soient A et B, $A\,\mathcal{R}\,B$ implique que pour tout contexte $\mathfrak{C}$, $\mathfrak{C}[A]\,\mathcal{R}\,\mathfrak{C}[B]$.*

Définition 1.6 (Congruence structurelle)**.** *On appelle* congruence structurelle *la plus petite relation d'équivalence contextuelle qui vérifie les règles de la figure 1.3.*

Fig. 1.4 Sémantique par réductions

$$
\begin{array}{rl}
\text{(R-COMM)} & l[\![a\,!\,\langle V\rangle\,P_1]\!] \mid l[\![a\,?\,(X:\mathsf{T})\,P_2]\!] \longrightarrow l[\![P_1]\!] \mid l[\![P_2\{^{V}/_{X}\}]\!] \\
\text{(R-GOTO)} & l[\![\mathsf{goto}\,k.\;P]\!] \longrightarrow k[\![P]\!] \\
\text{(R-IF-V)} & l[\![\mathsf{if}\;a=a\;\mathsf{then}\;P_1\;\mathsf{else}\;P_2]\!] \longrightarrow l[\![P_1]\!] \\
\text{(R-IF-F)} & l[\![\mathsf{if}\;a_1=a_2\;\mathsf{then}\;P_1\;\mathsf{else}\;P_2]\!] \longrightarrow l[\![P_2]\!] \quad \text{quand } a_1 \neq a_2 \\
\text{(R-NEWCHAN)} & l[\![\mathsf{newchan}\,c:\mathsf{C}\,\mathsf{in}\,P]\!] \longrightarrow (\mathsf{new}\,c:\mathsf{C}_{@l})\,l[\![P]\!] \\
\text{(R-NEWLOC)} & l[\![\mathsf{newloc}\,k,(\vec{c}):\mathsf{L}\,\mathsf{with}\,P_k\,\mathsf{in}\,P]\!] \longrightarrow (\mathsf{new}\langle(k,(\vec{c})):\mathsf{L}\rangle)\,k[\![P_k]\!] \mid l[\![P]\!] \\
\text{(R-SPLIT)} & l[\![P_1 \mid P_2]\!] \longrightarrow l[\![P_1]\!] \mid l[\![P_2]\!] \\
\text{(R-HERE)} & l[\![\mathsf{here}\,(x)\,P]\!] \longrightarrow l[\![P\{^{l}/_{x}\}]\!] \\
\text{(R-RÉP)} & l[\![*P]\!] \longrightarrow l[\![P]\!] \mid l[\![*P]\!] \\
\text{(R-REC)} & l[\![(\mathsf{rec}\;Z:\mathsf{R}\,(X)\,.\,P)\,\langle V\rangle]\!] \longrightarrow l[\![P\{^{V}/_{X}\}[^{\mathsf{rec}\;Z:\mathsf{R}(X).\;P}/_{Z}]]\!]
\end{array}
$$

$$\text{(R-C-PAR)} \quad \frac{M_1 \longrightarrow M_1'}{M_1 \mid M_2 \longrightarrow M_1' \mid M_2}$$

$$\text{(R-C-NEW)} \quad \frac{M_1 \longrightarrow M_1'}{(\mathsf{new}\,a:\mathsf{E})\,M_1 \longrightarrow (\mathsf{new}\,a:\mathsf{E})\,M_1'}$$

$$\text{(R-STRUCT)} \quad \frac{M_1 \equiv M_2 \longrightarrow M_2' \equiv M_1'}{M_1 \longrightarrow M_1'}$$

La règle de congruence structurelle qui permet effectivement la communication dans le système

$$l[\![a\,!\,\langle V\rangle\,P_1]\!] \mid (\mathsf{new}\,b:\mathsf{E}_b)\,l[\![a\,?\,(X:\mathsf{T})\,P_2]\!]$$

est celle qui récrit ce système sous la forme

$$(\mathsf{new}\,b:\mathsf{E}_b)(l[\![a\,!\,\langle V\rangle\,P_1]\!] \mid l[\![a\,?\,(X:\mathsf{T})\,P_2]\!])$$

par une extension de la portée du nom b, dès lors que le nom b n'est pas libre dans $l[\![a\,!\,\langle V\rangle\,P_1]\!]$. Les autres règles permettent de spécifier le fait que l'opérateur de composition | est à la fois associatif et commutatif, que le système **0** est totalement inactif et peut donc être supprimé d'un système sans en modifier l'activité, et enfin que l'ordre des portées de deux noms ne saurait modifier la sémantique du système représenté.

Nous pouvons enfin maintenant donner les règles de réduction qui fournissent le « cœur » de la sémantique. Ces règles sont données figure 1.4. On peut en distinguer deux sortes :

- les axiomes, comme (R-GOTO), qui ont simplement la forme d'une réduction $M \longrightarrow M'$ et qui indiquent que ces réductions sont toujours possibles ; ces axiomes correspondent à une vraie réduction du système, au cours de laquelle une primitive du langage agit ;

– les règles d'inférence, comme (R-STRUCT), qui ont la forme

$$\frac{M_1 \longrightarrow M_1'}{M \longrightarrow M'}$$

et qui permettent de conclure que la réduction $M \longrightarrow M'$ est possible dès lors que l'on peut prouver la réduction $M_1 \longrightarrow M_1'$.

Ces règles donnent la sémantique du calcul : le sens que nous donnerons à un système M est l'ensemble des réductions que ce système est capable d'effectuer, c'est-à-dire l'ensemble des $M \longrightarrow M'$ qui sont prouvables grâce à ces règles d'inférence. On parlera de *réduction faible*, notée $M \Longrightarrow M'$, quand il existe des systèmes $M_1, \ldots, M_n$ pour lesquels toutes les réductions $M_i \longrightarrow M_{i+1}$ sont prouvables, avec $M = M_1$ et $M' = M_n$. On la notera parfois $M \longrightarrow^* M'$ quand on s'intéressera plus à son aspect « suite de réductions ».

De façon plus générale, si $\mathcal{R}$ et $\mathcal{S}$ sont deux relations, par exemple $\longrightarrow$, on notera :

– $\mathcal{R}^*$ la fermeture réflexive et transitive de $\mathcal{R}$,
– $\mathcal{R}\mathcal{S}$ ou $\mathcal{R} \circ \mathcal{S}$ la composition de ces deux relations,
– $\mathcal{R}^+$ la relation $\mathcal{R} \circ (\mathcal{R}^*)$,
– $\mathcal{R}^=$ pour $\mathcal{R} \cup (=)$,
– $\mathcal{R}^n$ pour $\mathcal{R} \circ \cdots \circ \mathcal{R}$ où $\mathcal{R}$ est composée n fois,
– et enfin $\mathcal{R}^{-1}$ pour la relation symétrique de $\mathcal{R}$.

Les règles concernant la réduction des générateurs de nom, (R-NEWCHAN) et (R-NEWLOC), effectuent intuitivement l'allocation proprement dite du nom qu'elles créent. Au cours de ces étapes de calcul ces préfixes de processus sont consommés pour laisser place à la construction ($\mathsf{new}\, a : \mathsf{E}$) des systèmes. Alors que le préfixe $\mathsf{newchan}c : \mathsf{C}$ n'engendre qu'une seule construction ($\mathsf{new}\, c : \mathsf{C}_{@l}$), $\mathsf{newloc}k, (\vec{c}) : \mathsf{L}$ donne lieu à un ($\mathsf{new}\, k : \textsc{loc}$) suivi d'une suite de ($\mathsf{new}\, c_i : \mathsf{C}_{i@k}$). Pour effectuer cette étape, il est nécessaire de décomposer le type L pour obtenir le type de chaque nom créé. La notation $\langle (k, (\vec{c})) : \mathsf{L} \rangle$ représentera cette décomposition et sera définie formellement au chapitre suivant, la modification du type C en $\mathsf{C}_{@l}$ également.

La règle (R-C-PAR) de la sémantique justifie pleinement le fait que l'opérateur $|$ soit appelé « composition parallèle » : mis à part dans le cas où deux systèmes se réduisent de concert en procédant à une communication, les différents systèmes qu'il compose peuvent se réduire indépendamment des autres. Cette idée est centrale car on cherche à modéliser grâce à ce calcul des systèmes répartis. La sémantique fait cependant à cet égard une première hypothèse simplificatrice : bien qu'elle ne force pas l'ordre d'exécution entre deux réductions possibles à un « instant » donné, elle impose qu'une seule des deux ait lieu à cet « instant ». On parle de *sémantique par entrelacement* car les différentes réductions possibles sont exécutées séquentiellement.

Par ailleurs, (R-C-PAR) et (R-C-NEW) expriment sous forme de règles le fait que la sémantique par réductions est contextuelle : un contexte de la forme $[\cdot] \mid M$ ou $(\mathsf{new}\, a : \mathsf{E})[\cdot]$ ne bloque aucune réduction du système qui y est inséré.

La règle (R-COMM), quant à elle, impose strictement que deux processus soient dans la même localité pour communiquer ensemble. Les canaux situés dans deux localités distinctes ne sont ainsi liés en aucune manière. Afin d'éviter les ambiguïtés, les noms des canaux d'une localité seront toujours différents de ceux des autres localités. Cette contrainte sera par ailleurs imposée par le

typage que nous verrons dans la suite. Ce choix a cependant un impact direct sur les processus récursifs : pour qu'un tel processus puisse s'exécuter dans des localités distinctes au cours de ses réductions, il devra apprendre non seulement les localités où migrer mais également les noms des canaux qu'il peut utiliser. Les paramètres des processus récursifs seront particulièrement utiles pour cela.

Enfin la règle (R-STRUCT) indique de quelle façon la congruence structurelle intervient dans la définition de la sémantique des systèmes : si deux systèmes sont structurellement congrus, cette règle transforme toute réduction de l'un en une réduction de l'autre. Les autres règles correspondent assez fidèlement aux intuitions apparues dans la présentation informelle du calcul.

Avant de poursuivre, maintenant que la sémantique du Dπ-calcul est définie, nous devons vérifier qu'il est effectivement possible de toujours supposer les systèmes clos.

Proposition 1.7 (Préservation de la clôture)**.** *Si le système M est clos et que la réduction $M \longrightarrow M'$ est prouvable alors M' est clos.*

Démonstration. Cette proposition se prouve très facilement par induction sur la preuve de la réduction $M \longrightarrow M'$. En effet, seules deux règles réduisent des lieurs de variables, (R-COMM) et (R-REC), et les variables « libérées » sont immédiatement substituées. □

Cette hypothèse de clôture est indispensable pour que les règles de réduction des tests aient bien le sens voulu. En effet, au moment où le processus $\mathsf{if}\ u_1 = u_2\ \mathsf{then}\ P\ \mathsf{else}\ Q$ se réduit, u_1 et u_2 doivent être des noms. Alors que les variables peuvent être substituées, les noms sont « constants », tester leur égalité a donc un sens bien défini.

Bien que la seule opération que l'on autorise sur les valeurs du langage soit le test d'égalité-inégalité, il sera très utile dans la suite de tester l'égalité de valeurs plus complexes qu'un simple nom. En particulier, après réception d'un message qui peut être un n-uplet, on voudra vérifier qu'il contient exactement les valeurs que l'on attend. De façon similaire, nous utiliserons des substitutions plus complexes que la simple $\{^v/_x\}$. Nous avons en effet déjà rencontré des substitutions de la forme $\{^V/_X\}$ dans la règle (R-COMM) de la sémantique. Nous lui donnons maintenant sa signification complète. Formalisons pour cela la décomposition des valeurs pour ramener une égalité $V_1 = V_2$ à une suite d'égalités $u_{11} = u_{21}, \ldots, u_{1n} = u_{2n}$. Nous allons même considérer en toute généralité des valeurs étendues.

Définition 1.8 (Décomposition arborescente simultanée)**.** *Pour deux valeurs étendues*[8] *S et T, la décomposition arborescente simultanée de (S, T) est la liste définie ainsi :*

- *l'unique couple (s, t) quand $S = s$ et $T = t$;*
- *la concaténation des décompositions des (S_i, T_i) si $S = (S_1, \ldots, S_n)$ et $T = (T_1, \ldots, T_n)$ pour un n quelconque.*

La décomposition n'est pas définie pour des valeurs qui ne partagent pas la même structure arborescente.

Quand $(s_1, t_1), \ldots, (s_n, t_n)$ est la décomposition arborescente simultanée de la paire (S, T), la substitution $\{^T/_S\}$ correspond à l'enchaînement des substitu-

[8] En particulier, S et T peuvent être des motifs.

tions $\{^{t_1}/_{s_1}\}$ à $\{^{t_n}/_{s_n}\}$. De façon similaire, le processus

$$\mathsf{if}\ S = T\ \mathsf{then}\ P\ \mathsf{else}\ Q$$

se résume à la cascade de tests

$$\mathsf{if}\ s_1 = t_1\ \mathsf{then}\ \mathsf{if}\ s_2 = t_2\ \mathsf{then}\ \ldots \mathsf{if}\ s_n = t_n\ \mathsf{then}\ P\ \mathsf{else}\ Q \ldots\ \mathsf{else}\ Q\ \mathsf{else}\ Q$$

Pour que cette décomposition fonctionne bien il faut donc que la décomposition arborescente simultanée de (S, T) soit définie, c'est-à-dire que S et T partagent la même structure arborescente. Le typage introduit dans le chapitre suivant pourra permettre d'assurer ce genre de propriétés.

Maintenant que nous disposons d'une sémantique bien définie, nous allons formaliser dans quel cas deux systèmes se comportent de façon identique.

1.3 Équivalence observationnelle

Modéliser une application dans un formalisme donné prend tout son sens lorsqu'on peut retirer du modèle une meilleure compréhension de l'application. Une des façons de mieux la comprendre, celle à laquelle nous allons particulièrement nous attacher dans cette thèse, est de pouvoir utiliser une notion précise d'équivalence existant dans le formalisme. Cette équivalence peut alors servir à vérifier l'adéquation entre l'application et sa spécification, lorsque toutes deux sont exprimées dans le formalisme. La littérature traitant de cette approche est riche, par exemple [LM87], [Par87], [Wal89] ou encore [BH00]. Bien entendu, pour qu'une telle approche du problème de vérification soit fructueuse, il faut que la spécification soit exprimée de façon suffisamment simple, même lorsque la modélisation de l'application proprement dite est bien plus difficile à appréhender. L'équivalence qui sera utile devra donc discerner subtilement les aspects des modèles qui doivent être distingués de ceux qui peuvent sans risques être confondus.

Par ailleurs il est très facile de définir une multitude d'équivalences totalement ou subtilement distinctes les unes des autres. Pour qu'une équivalence ne soit pas purement arbitraire, il est essentiel de bien justifier ses différents aspects, les choix qui mènent à sa définition particulière. C'est pourquoi les choix qui régissent la définition que nous allons voir sont façonnés par leurs intuitions.

Notre équivalence sera donc *observationnelle*. Cela signifie que le critère retenu pour affirmer que deux systèmes sont non-équivalents sera la possibilité pour un *observateur* de faire la différence entre ces deux systèmes. En effet, si en interagissant avec l'un ou l'autre de ces deux systèmes, tout se déroule exactement de la même manière, on devrait pouvoir remplacer l'un par l'autre. L'intuition qui suggère ce choix repose sur la simple idée qu'il importe peu à l'utilisateur d'un serveur de savoir comment sont effectués les calculs si le comportement qu'il constate est identique. Cette intuition est très proche de la notion d'égalité extensionnelle pour les fonctions mathématiques : quelle que soit la formule précise utilisée pour le calcul, on les prend pour égales dès qu'on ne peut les différencier en observant leur valeur en un point donné. Les systèmes seront donc considérés comme des boîtes noires, l'observateur devra se contenter d'interagir avec elles pour les différencier.

Forts de cette intuition, il nous faut maintenant décrire les principes qui régissent l'observation :

1. il pourra voir si la boîte noire peut émettre un message sur un canal libre car, lors d'une telle émission, le système interroge en quelque sorte son environnement, en particulier l'observateur de la boîte noire, pour savoir s'il accepte de communiquer avec lui sur ce canal ;
2. il pourra placer la boîte noire dans un plus grand système de son choix et observer le résultat ;
3. par contre, il ne pourra pas distinguer si la boîte noire effectue des calculs, dans la mesure où cela relève de la façon particulière dont le système implémente ses fonctionnalités, c'est-à-dire les interactions qu'il peut engager avec son environnement ; pour poursuivre la métaphore des fonctions mathématiques, on cherchera à confondre les fonctions $x \mapsto 0$ et $x \mapsto x! - x!$.

Formalisons la notion d'équivalence qui découle de ces droits et contraintes. Pour le droit numéro 1, nous appellerons *barbe* un tel message.

Définition 1.9 (Barbe)**.** *Le système M exhibe une barbe sur le canal a dans la localité l si et seulement s'il existe un processus P et un système M' tels que*

$$M \Longrightarrow\equiv (\mathsf{new}\,\vec{n} : \vec{\mathsf{E}})(M' \mid l[\![a\,!\,\langle V\rangle\, P]\!])$$

avec $a, l \notin \vec{n}$.

On notera l'exhibition d'une telle barbe $M \Downarrow a_{@}l$.

Cette définition d'une barbe découle tout simplement du fait qu'un système pouvant émettre un message doit être de la forme

$$(\mathsf{new}\,\vec{n}_{(1)} : \vec{\mathsf{E}}_{(1)})(M_{(1)} \mid (\mathsf{new}\,\vec{n}_{(2)} : \vec{\mathsf{E}}_{(2)})(M_{(2)} \mid \ldots l[\![a\,!\,\langle V\rangle\, P]\!] \ldots \mid N_{(2)}) \mid N_{(1)})$$

ce qui est structurellement équivalent à un système de la forme bien plus simple

$$(\mathsf{new}\,\vec{n} : \vec{\mathsf{E}})(M' \mid l[\![a\,!\,\langle V\rangle\, P]\!])$$

Quant au fait que l'on autorise au passage un nombre arbitraire de réductions avant que ne soit atteint cet état du système, cela retranscrit la contrainte numéro 3 qui empêche l'observateur de comptabiliser les réductions du système observé.

Le droit numéro 1 peut alors s'exprimer comme une contrainte sur les systèmes qu'une équivalence peut mettre en relation.

Définition 1.10 (Respect des barbes[9])**.** *On dit qu'une relation $\mathcal{R}$ respecte les barbes quand $M\ \mathcal{R}\ N$ implique que, quels que soient a et l, $M \Downarrow a_{@}l$ si et seulement si $N \Downarrow a_{@}l$.*

Le droit numéro 2 que nous accordons à l'observateur implique que la relation d'équivalence que nous cherchons doit être contextuelle, puisque placer un système dans un autre plus grand, c'est le placer dans un contexte.

Enfin, la contrainte numéro 3 se formalise également facilement sous la forme d'une propriété de la relation d'équivalence.

Définition 1.11 (Fermeture par réduction)**.** *On dit qu'une relation $\mathcal{R}$ est* fermée par réduction *quand $M\ \mathcal{R}\ N$ implique que pour toute réduction $M \longrightarrow M'$ il existe une réduction faible $N \Longrightarrow N'$ telle que $M'\ \mathcal{R}\ N'$.*

[9] Et des barbus

Nous avons maintenant tous les outils pour définir la relation d'équivalence décrite informellement.

Définition 1.12 (Congruence barbue fermée par réduction)**.** *La* congruence barbue fermée par réduction *est la plus grande relation symétrique contextuelle, fermée par réduction et qui respecte les barbes. On la note* $\cong^{br}$*.*

Bien entendu, la première propriété intéressante à propos de cette relation est le fait qu'il s'agit bien d'une relation d'équivalence, comme promis. $\cong^{br}$ est définie comme la plus grande relation vérifiant ses hypothèses de définition, elle contient donc toutes les autres relations qui vérifient ces mêmes hypothèses. Il suffira ainsi, pour démontrer qu'une relation est contenue dans $\cong^{br}$, de prouver qu'elle vérifie ses hypothèses de définition.

Proposition 1.13. *La congruence barbue fermée par réduction est une relation d'équivalence.*

Démonstration. Il suffit pour cela de montrer que cette relation doit être réflexive et transitive car on sait déjà qu'elle est symétrique.

Pour prouver qu'elle est réflexive, il suffit de voir que la relation identité, c'est-à-dire contenant tous les couples (M, M) pour tout système M, vérifie les hypothèses de définition de $\cong^{br}$.

Pour prouver sa transitivité, on peut constater assez facilement que la composition de $\cong^{br}$ avec elle-même, c'est-à-dire $\cong^{br} \circ \cong^{br}$, vérifie toujours ses hypothèses de définition. En effet, si (M_1, M_3) est un couple de $\cong^{br} \circ \cong^{br}$, il doit exister M_2 tel que $M_1 \cong^{br} M_2$ et $M_2 \cong^{br} M_3$. Par symétrie de $\cong^{br}$, on obtient $M_3 \cong^{br} M_2$ et $M_2 \cong^{br} M_1$ soit $(M_3, M_1) \in (\cong^{br} \circ \cong^{br})$. Le raisonnement est identique pour la contextualité et le respect des barbes. Pour la fermeture par réduction, $M_1 \longrightarrow M_1'$ implique que $M_2 \Longrightarrow M_2'$, c'est-à-dire $M_2 \longrightarrow^n M_2'$. Une simple induction sur n donne $M_3 \Longrightarrow^n M_3'$, ce qui achève la preuve. □

Armés de cette équivalence et étant donnés deux systèmes, nous voudrions maintenant pouvoir prouver qu'ils sont équivalents ou qu'ils ne le sont pas. La preuve que nous venons de voir illustre une méthodologie pour prouver qu'un couple de systèmes est dans $\cong^{br}$: on choisit une « bonne » relation contenant le couple, on prouve que cette relation vérifie les hypothèses de définition de $\cong^{br}$ et on en déduit que cette relation, en particulier le couple de systèmes auquel on s'intéresse, est inclus dans $\cong^{br}$.

Exemple 1.14. Considérons un exemple d'équivalents, en omettant les annotations de types qui ne sont pas encore expliquées et ne jouent de toute façon aucun rôle pour la congruence $\cong^{br}$. Nous allons voir comment se passerait la preuve que le système $M = (\mathsf{new}\, a)\, l[\![a\,!\,\langle\rangle]\!] \mid l[\![a\,?\,()]\!]$ est équivalent au simple système $\mathbf{0}$. En procédant suivant la méthode que nous venons d'esquisser, il faut exhiber une relation contenant $(M, \mathbf{0})$ et vérifiant les caractéristiques de $\cong^{br}$. Essayons de construire cette relation $\mathcal{R}$ en partant de ce simple couple. Pour rendre $\mathcal{R}$ symétrique, rajoutons simplement $(\mathbf{0}, M)$. Pour tous les éléments que nous rajouterons dans la relation par la suite, nous veillerons à toujours y rajouter également le symétrique. Comme M et $\mathbf{0}$ n'exhibent pas la moindre barbe ni l'un ni l'autre, il n'y a pas d'effort particulier à faire pour cela. Pour fermer notre relation par réduction, la tâche est encore assez facile car les seules réductions pouvant se faire sont celles menant M à $M' = (\mathsf{new}\, a)\, l[\![\mathsf{stop}]\!] \mid l[\![\mathsf{stop}]\!]$ et aux systèmes qui lui sont

structurellement congrus, par exemple $((\mathsf{new}\, a)\, l[\![\mathsf{stop}]\!]) \mid l[\![\mathsf{stop}]\!]$. Il nous suffit donc de rajouter à $\mathcal{R}$ les couples $(M', \mathbf{0})$ et $(\mathbf{0}, M')$ et les couples correspondant pour les autres systèmes structurellement congrus à M'. Malheureusement, il faut encore prendre en compte la contextualité, donc passer de notre petite relation à la relation constituée des couples $(\mathfrak{C}[M_1], \mathfrak{C}[M_2])$ où (M_1, M_2) est dans notre relation et $\mathfrak{C}$ est un contexte quelconque. Il nous faut maintenant vérifier que cette relation $\mathcal{R}$ vérifie toujours les autres propriétés. Elle est bien sûr toujours symétrique et elle respecte toujours les barbes. Celles-ci sont en effet toujours dues au contexte qui est identique pour les deux composantes de chaque paire, puisque nos deux systèmes initiaux n'en exhibaient aucune. Par contre, la preuve que, lorsque $\mathfrak{C}[M_1]$ effectue une étape de calcul, le résultat doit forcément être structurellement équivalent à un $\mathfrak{C}'[M_1']$ où M_1' est l'un des trois M, M' ou $\mathbf{0}$, est bien plus technique. La preuve que $\mathcal{R}$ est effectivement fermée par réduction découle alors de l'imitation de cette réduction par une réduction $\mathfrak{C}[M_2] \Longrightarrow \mathfrak{C}'[M_2']$ où (M_1', M_2') est l'un des quatre couples $(M, \mathbf{0})$, $(\mathbf{0}, M)$, $(M', \mathbf{0})$ ou $(\mathbf{0}, M')$. Alors seulement nous pouvons déduire que $\mathcal{R}$ est incluse dans $\cong^{br}$.

Bien que nous ayons maintenant une notion d'équivalence sur les systèmes, le petit exemple de preuve que nous venons d'esquisser montre à quel point l'équivalence est dure à prouver même dans un cas très simple. Quand nous introduirons dans la suite des variantes typées de cette congruence, nous verrons une formulation alternative de l'équivalence plus adaptée aux preuves et plus propice à comprendre les raisons en cas de non-équivalence.

Chapitre 2

Typage

2.1 Misère du calcul sans type

Voyons sur des exemples quelques problèmes que le typage permettra de traiter.

La sémantique du Dπ-calcul définie au chapitre précédent affirme que le système

$$l[\![a\,!\,\langle V\rangle\,P]\!] \mid l[\![a\,?\,(X : \mathsf{T})\,Q]\!]$$

peut se réduire, en une étape de calcul, au système

$$l[\![P]\!] \mid l[\![Q\{^{V}/_{X}\}]\!]$$

où la valeur V et le motif X ne sont pas spécifiés avec précision. Par ailleurs, nous avons vu comment définir cette substitution $\{^{V}/_{X}\}$ en la décomposant en une suite de substitutions élémentaires. Mais cette substitution n'est définie que lorsque X et V partagent la même structure arborescente. Dans le cas contraire, on aboutit à une expression de la forme

$$l[\![a\,!\,\langle(b, c)\rangle\,P]\!] \mid l[\![a\,?\,((x, y, z) : \mathsf{T})\,Q]\!]$$

qui n'a guère de signification intuitive. De plus, la sémantique ne permet pas de l'interpréter. Nous chercherons alors à assurer que toutes les utilisations d'un canal de communication mettent en jeu des messages de même structure.

D'autre part on peut très facilement construire un système tout aussi problématique pour une raison plus subtile. Considérons donc le système

$$k[\![c\,?\,()\,P]\!] \mid l[\![a\,!\,\langle k, c\rangle]\!] \mid l[\![a\,?\,((x, y) : \mathsf{T})\,\mathsf{goto}\,y.\ x\,!\,\langle\rangle]\!]$$

où le deuxième processus avertit le troisième que le canal sur lequel il doit émettre un message est le canal c dans la localité k. Mais, tous les noms du calcul étant confondus, cette information est interprétée par le troisième processus comme étant le canal k dans la localité c. En effet la syntaxe ne différencie pas les localités des canaux. Qui plus est, elle ne permet pas d'indiquer que le canal c est situé dans la localité k et non ailleurs car, comme nous l'avons déjà signalé, un canal n'existe que dans une localité précise. Nous nous efforcerons donc de différencier les localités des canaux et de fournir un moyen de spécifier dans quelle localité est disponible quel canal.

Bien entendu, les deux problèmes précédents sont moins évidents à détecter quand les communications ont lieu sur des canaux dont les noms ont été reçus dynamiquement. Par exemple le système

$$l[\![d\,!\,\langle a\rangle\,e\,!\,\langle a\rangle]\!] \mid l[\![d\,?\,(x_1)\,x_1\,!\,\langle(b,c)\rangle\,P]\!] \mid l[\![e\,?\,(x_2)\,x_2\,?\,((y_1,y_2,y_3):\mathsf{T})\,Q]\!]$$

laisse apparaître après deux communications, sur les canaux d et d', le problème des tailles différentes de message sur le canal a. Alors que les deux exemples précédents pouvaient laisser envisager une solution très simple qui associerait à chaque nom soit le statut de localité soit celui de canal apte à transmettre des messages d'une taille donnée, cet exemple illustre le fait qu'il faille également, dans le cas des canaux, indiquer entièrement le statut qu'ont les données qui y transitent.

Ces problèmes rappellent fortement ceux d'un langage de programmation : un nombre incorrect d'arguments lors d'un appel de fonction, une chaîne de caractères confondue avec un entier ou encore l'utilisation de fonctions d'ordre supérieur. Une approche souvent suivie pour leur apporter une solution, au moins pour les cas les plus simples, est de rajouter du typage au langage. Nous allons aussi suivre cette voie en associant à chaque nom et chaque variable un *type*, c'est-à-dire une information indiquant si l'identifiant est une localité ou un canal et, si c'est un canal, les types des valeurs pouvant y être échangées. Cette association permettra de vérifier qu'un processus ou un système utilise toujours ses noms et variables de façon légitime, dans des circonstances conformes au type associé. De plus, cette vérification pourra être effectuée par une analyse statique, c'est-à-dire avant l'exécution du système ; ainsi elle n'entravera pas son exécution tout en garantissant qu'aucune erreur ne sera commise au cours du calcul.

Deux autres exemples dicteront des choix importants quant aux types effectivement introduits par la suite. Tout d'abord, revenons sur le processus-prospecteur présenté à la section 1.1.

$$\begin{array}{l} l\ [\![\\ \quad (\\ \quad\quad \mathsf{rec}\ Z : \mathsf{R}\,(x_{val}, x_{vois})\,. \\ \quad\quad\quad x_{val}\,?\,(x:\mathsf{T}) \\ \quad\quad\quad \mathsf{if}\ x = or \\ \quad\quad\quad \mathsf{then\ here}\,(y)\ \mathsf{goto}\ campement.\ locOr\,!\,\langle y\rangle \\ \quad\quad\quad \mathsf{else}\ x_{vois}\,?\,((y,X):\mathsf{T}')\ \mathsf{goto}\ y.\ Z\,\langle X\rangle \\ \quad)\,\langle val, vois\rangle \\ ]\!] \end{array}$$

Considérons le type que l'on pourrait associer à la variable x_{vois}. Sur ce canal, on attend un couple composé d'une localité (y) et des arguments nécessaires à l'appel récursif (X). Or les arguments de l'appel récursif sont les deux canaux x_{val} et, surtout, x_{vois} lui-même, comme l'indique la déclaration du processus récursif. Suivant les conclusions auxquelles nous étions arrivés quant aux informations que doivent véhiculer les types, le type du canal x_{vois} devrait donc mentionner son propre type. C'est pourquoi nous utiliserons des types récursifs, fort utiles pour parler de processus récursifs aussi bien que d'autres comportements infinis décrits en utilisant la réplication, l'autre construction fournie par le calcul autorisant la répétition.

Enfin, le dernier aspect important du typage permettra de contrôler l'usage qui est fait des ressources. Imaginons par exemple un serveur de calcul, exprimé par un processus répliqué de la forme

$$l[\![*req\,?\,(X : \mathsf{T})\,P]\!]$$

autorisant d'autres processus à lui envoyer des requêtes. La mise en place d'un tel service suppose la création de ce processus répliqué ainsi que la diffusion du nom du canal *req* sur lequel lancer les requêtes. Cela pourrait par exemple prendre la forme suivante :

$$l[\![*req\,?\,(X : \mathsf{T})\,P \mid *nouv\text{-}service\,!\,\langle req\rangle]\!]$$

La définition même du service suppose que tous les clients qui recevront ainsi le canal *req* n'utiliseront que le droit d'y émettre des messages. Mais il est possible d'envisager un processus-usurpateur

$$l[\![nouv\text{-}service\,?\,(y : \mathsf{T}')\,y\,?\,(X : \mathsf{T})\,P']\!]$$

qui reçoit le nom du canal servant à émettre des requêtes et en détourne une, en remplaçant le comportement du service par P'. Nous pourrons empêcher ce détournement en distinguant les capacités d'émission et de réception sur les canaux et en ne transmettant que la capacité d'émettre sur le canal *req* lorsque le nom est diffusé sur le canal *nouv-service*.

2.2 Syntaxe des types

Suivant les intuitions naissant des exemples précédents, on attribuera dans la suite à chaque identifiant un type qui différenciera les localités des canaux. Pour une localité, ce type sera simplement LOC. Par contre pour un canal, le type comportera à la fois trois informations :

- les capacités disponibles pour ce canal, que ce soit l'émission, la réception ou leur combinaison ; ces capacités indiquent ainsi ce qu'il est possible de faire avec le canal ;
- les types des valeurs échangées sur ce canal ;
- la localité dans laquelle ce canal existe, puisqu'un canal est lié à une unique localité.

En combinant ces trois éléments, on associera donc le type suivant à un canal situé dans la localité l sur lequel on a le droit de recevoir des valeurs de type T :

$$\mathrm{R}\langle \mathsf{T}\rangle_{@l} \quad \text{avec les significations} \quad \begin{array}{lcl} \mathrm{R} & : & \text{capacité de réception} \\ \mathsf{T} & : & \text{type des valeurs reçues} \\ l & : & \text{localisation du canal} \end{array}$$

Prenons l'exemple simple d'un canal de synchronisation. Sur un tel canal, les messages véhiculés sont vides. La seule information communiquée est donc la présence du message, ce qui suffit à synchroniser deux processus s'exécutant en parallèle. On associera donc un type $\mathrm{R}\langle\rangle_{@l}$ à un tel canal. Si c est un canal de type $\mathrm{R}\langle\rangle_{@l}$, un processus s'exécutant dans l pourra recevoir un message sur c de la façon suivante : $l[\![c\,?\,()]\!]$. Par contre, les systèmes suivants ne respectent pas l'information de typage :

– $l[\![c\,!\,\langle\rangle]\!]$ car le processus tente d'émettre un message alors qu'il ne dispose que de la capacité de réception ;
– $k[\![c\,?\,()]\!]$ car c n'existe que dans la localité l ;
– $l[\![c\,?\,(x:\mathsf{T})]\!]$ car les messages circulant sur c sont vides.

De façon similaire, le type correspondant à la capacité d'émettre des valeurs de type T dans la même localité l sera :

$\textsc{w}\langle\mathsf{T}\rangle@l$ avec les significations

$\textsc{w}$	:	capacité d'émission
T	:	type auquel les valeurs sont émises
l	:	localisation du canal

Assez naturellement, la combinaison des deux capacités d'émission et réception pourra prendre la forme d'un type $\textsc{rw}\langle\mathsf{T}\rangle@l$. Mais il est important pour le typage d'offrir des combinaisons plus souples de ces capacités. Pour présenter cette subtilité, commençons par considérer deux canaux c et d de types respectifs $\textsc{rw}\langle\rangle@l$ et $\textsc{rw}\langle\textsc{r}\langle\rangle\rangle@l$. Les canaux qui circulent sur le canal d de la localité l sont donc aussi situés dans l et la seule capacité communiquée est la réception de messages vides. Dans la mesure où les deux capacités d'émission et réception sont disponibles pour le canal c, l'intuition veut que l'on puisse se restreindre à n'utiliser que la capacité de réception et à l'envoyer sur le canal d. Ainsi, le système $l[\![d\,!\,\langle c\rangle]\!]$ respecte les contraintes des types de c et d. Ce rapport qui existe entre les types $\textsc{rw}\langle\rangle@l$ et $\textsc{r}\langle\rangle@l$ est un cas particulier d'une notion plus générale : quand toutes les valeurs qui ont le type T_1 peuvent être utilisées là où une valeur de type T_2 est attendue, on dira que T_1 est un *sous-type* de T_2. Cette notion sera définie plus formellement dans la suite, mais elle devient rapidement moins intuitive quand les types s'emboîtent.

On se concentre donc sur l'intuition du système de types en cours d'élaboration : les communications ne permettent pas uniquement la transmission des noms (ou variables) mais aussi et surtout des capacités associées à ces identifiants. Par conséquent, on voudra pouvoir parler du type auquel un processus connaît un canal si ce processus a acquis des capacités différentes en réception et en émission pour ce canal. On autorise donc que les types en réception et émission soient distincts dans les types de canaux. Considérons un cas où cet ajustement s'avère intéressant : $\textsc{rw}\langle\textsc{rw}\langle\rangle\rangle@l$ est le type d'un canal de la localité l sur lequel envoyer et recevoir des canaux situés, eux aussi, dans l et véhiculant des messages vides. On peut aisément se convaincre que ce type est un sous-type de $\textsc{w}\langle\textsc{rw}\langle\rangle\rangle@l$, car il suffit simplement d'« oublier » la capacité de réception. Il est moins évident de voir que c'est aussi un sous-type de $\textsc{r}\langle\textsc{r}\langle\rangle\rangle@l$. Mais si c est un canal de type $\textsc{rw}\langle\textsc{rw}\langle\rangle\rangle@l$, on peut oublier la capacité d'émission à la fois sur c lui-même et sur tous les canaux qui sont reçus sur c.

Un processus pourra donc recevoir c successivement aux types $\textsc{w}\langle\textsc{rw}\langle\rangle\rangle@l$ et $\textsc{r}\langle\textsc{r}\langle\rangle\rangle@l$. On notera $\textsc{rw}\langle\textsc{r}\langle\rangle,\textsc{rw}\langle\rangle\rangle@l$ pour cette combinaison particulière de capacités. Ces combinaisons devront cependant être utilisées avec soin pour que le typage permette effectivement d'éviter les erreurs d'exécution : il faudra contraindre les valeurs qui sont émises sur un canal (donc ayant le type en émission) à pouvoir être reçues et utilisées sans risques (donc ayant aussi le type en réception). Pour cela, le système garantira que les types en émission sont des sous-types de ceux en réception, sur tous les types de canaux. Dans l'exemple précédent, cette contrainte est effectivement vérifiée puisque le type $\textsc{rw}\langle\rangle$ est bien un sous-type de $\textsc{r}\langle\rangle$.

Fig. 2.1 Pré-types

$\mathsf{C} ::=$	*Types des canaux locaux*
$\text{R}\langle \mathsf{T}_1 \rangle$	Lecture de valeur de type T_1
$\text{W}\langle \mathsf{T}_2 \rangle$	Écriture de valeur de type T_2
$\text{RW}\langle \mathsf{T}_1, \mathsf{T}_2 \rangle$	Intersection des deux types précédents
$\mathsf{E} ::=$	*Types des identifiants*
LOC	Localité
$\mathsf{C}_{@s}$	Canal localisé à s
$\mathsf{T} ::=$	*Types des valeurs transmissibles*
C	Canal local
E	Identifiant
$(\mathsf{T}_1, \ldots, \mathsf{T}_n)$	n-uplet
$\sum \vec{x} : \vec{\text{LOC}}.\ \mathsf{T}$	Somme dépendante
$\text{REC}\, \mathbf{Y}.\ \mathsf{T}$	Type récursif
$\mathbf{Y}$	Variable récursive
$\mathsf{L} ::=$	*Types déclaratifs de localité*
$\sum x : \text{LOC}.\ (\mathsf{C}_{1@x}, \ldots, \mathsf{C}_{n@x})$	
$\mathsf{R} ::=$	*Types déclaratifs de récursion*
$\sum x : \text{LOC}.\ \mathsf{T}$	

La syntaxe des types est définie formellement à la figure 2.1. Bien que les types de canaux puissent combiner des types distincts en réception et émission, on abrégera en $\text{RW}\langle \mathsf{T} \rangle$ les types des canaux sur lesquels ces deux types sont tous deux égaux à T. On distinguera plusieurs familles de types en utilisant explicitement des lettres différentes. E désignera ainsi les types possibles pour un nom ou une variable, c'est-à-dire un type de localité ou un type de canal localisé. C correspondra quant à lui aux types possibles pour les canaux locaux, c'est-à-dire en omettant simplement la localité dans laquelle il est situé. Comme l'indiquait la syntaxe du calcul (figure 1.2), le type assigné à un canal à sa création est un type de la forme C. La syntaxe impose par cette notation que le type de la construction $\mathsf{newchan}\, c : \mathsf{C}\, \mathsf{in}\, P$ ne soit pas explicitement localisé et, par conséquent, que seuls des processus s'exécutant dans la localité l puisse y créer des canaux. Ce choix est caractéristique de la vision de la répartition du calcul que porte le Dπ typé.

Les types des valeurs transmissibles seront dénotés T. Les trois constructions essentielles que ces types introduisent sont : le n-uplet, la somme dépendante et le type récursif. Le n-uplet est un produit cartésien habituel. La somme dépendante [ML84] est aussi une forme de produit cartésien : ce type pourra être associé à une paire (l, c) dans laquelle le type de c mentionne le nom l. Cette somme dépendante a une forme très simple ici : puisque les seuls identifiants pouvant apparaître dans les types sont des localités, la première composante de la somme dépendante est un n-uplet de localités. Elle apporte la possibilité de décrire la co-localisation de canaux. Ainsi, attribuer le type $\sum x : \text{LOC}.\ \mathsf{C}_{1@x}, \ldots, \mathsf{C}_{n@x}$ à une valeur $(l, (c_1, \ldots, c_n))$ permet d'indiquer que les canaux (c_i) sont tous localisés dans la même localité l. Qui plus est, les types n-uplet et somme

dépendante imposent la structure arborescente des valeurs, ce qui permet de traiter une difficulté soulevée dans la section 2.1.

Les types récursifs sont décrits par un opérateur de plus petit point fixe standard noté $\text{REC}\,\mathbf{Y}.\,\mathsf{T}$. Par ailleurs ces deux dernières constructions de type, le type récursif $\text{REC}\,\mathbf{Y}.\,\mathsf{T}$ et la somme dépendante $\sum \vec{x} : \vec{\text{LOC}}.\,\mathsf{T}$, sont des lieurs pour leur variable respective $\mathbf{Y}$ et $\vec{x}$. On leur appliquera donc des α-conversions et la même convention de Barendregt (convention 1.3) qu'aux lieurs du Dπ-calcul proprement dit. De ces lieurs dans les types découlent encore une fois naturellement des notions de noms et variables libres. Les figures A.1 et A.2 de l'annexe A formalisent aussi ces notions pour les types.

La figure 2.1 définit encore deux familles de types déclaratifs, à savoir ceux de localité et ceux de récursion. Ces types sont uniquement des cas particuliers des valeurs transmissibles et servent donc essentiellement de notation pour des types précis. On parle de types *déclaratifs* pour mettre en avant le fait que ces types n'apparaissent dans le langage qu'à la déclaration d'une nouvelle localité, avec la primitive $\mathsf{newloc}\,k, (\vec{c}) : \mathsf{L}\,\mathsf{with}\,P_k\,\mathsf{in}\,P$, ou d'un processus récursif, avec la primitive $(\mathsf{rec}\ Z : \mathsf{R}\,(X)\,.\ P)\,\langle V\rangle$. La syntaxe particulière des types déclaratifs de localité, à savoir $\sum x : \text{LOC}.\,(\mathsf{C}_1{}_{@}x, \ldots, \mathsf{C}_n{}_{@}x)$, utilise une somme dépendante pour forcer les canaux $\vec{c}$ à être localisés dans k. Pour leur part, les types déclaratifs de récursion R serviront à typer aussi bien le paramètre X que l'argument V. Ce typage particulier sera détaillé avec le typage des processus récursifs, à la section 2.7.

La figure 2.1 définit uniquement les *pré-types*. On parle de pré-types afin d'insister sur l'ajout de contraintes sur cette syntaxe pour obtenir les types proprement dits. La première contrainte portant sur les pré-types concernera leur clôture. On supposera ainsi les variables de récursion $\mathbf{Y}$ des types toujours liées. On imposera aussi que les types de la forme C ou R soient totalement clos, c'est-à-dire sans nom ni variable libre. Cette hypothèse s'avérera particulièrement importante pour certaines propriétés du typage[1]. Une telle contrainte ne décimera pas l'ensemble des processus descriptibles dans le calcul dans la mesure où il est possible de transmettre un canal c de type $\mathsf{C}_{@}s$ malgré tout : il suffira, en effet, de former la valeur (s, c), par exemple, avec le type dépendant $\sum x : \text{LOC}.\,\mathsf{C}_{@}x$.

La deuxième contrainte que l'on utilisera concerne les points fixes pouvant apparaître dans les pré-types. Par exemple, le type $\text{REC}\,\mathbf{Y}.\,\mathbf{Y}$ indique uniquement qu'il est égal à lui-même, sans plus de précisions sur les valeurs ayant ce type. En effet, on utilise le récurseur $\text{REC}\,\mathbf{Y}$ de façon standard : le type récursif $\text{REC}\,\mathbf{Y}.\,\mathsf{T}$ correspond au type T dans lequel tous les $\mathbf{Y}$ sont remplacés par le type récursif complet $\text{REC}\,\mathbf{Y}.\,\mathsf{T}$ à nouveau. Ainsi le type $\text{REC}\,\mathbf{Y}.\,\text{R}\langle\mathbf{Y}\rangle$ correspond à un canal sur lequel on peut lire des canaux du même type, chaque substitution de $\mathbf{Y}$ précisant le type. Mais dans le cas du type $\text{REC}\,\mathbf{Y}.\,\mathbf{Y}$ cette substitution n'apporte aucune information supplémentaire puisque le type $\mathbf{Y}$ ne contient aucun constructeur de type tel que $\text{R}\langle\cdot\rangle$. Le problème se pose de la même manière pour un type $\text{REC}\,\mathbf{Y}.\,\text{REC}\,\mathbf{Y}'.\,\mathbf{Y}$. On interdira donc de tels types en imposant

[1] Le fait que les types de canaux locaux C soient totalement clos garantit que les seuls noms pouvant apparaître dans le type d'un identifiant sont des localités. Grâce aux sommes dépendantes, cela implique qu'il est possible d'associer à toute valeur un type clos, quitte à rajouter des noms de localités et des dépendances de type. Il est en particulier possible d'associer une valeur et un type à tout environnement (voir définition 4.48). Cet aspect est fondamental dans la preuve de coïncidence des équivalences. Quant aux types R, les appels récursifs sont, d'une certaine façon, des communications (voir chapitre 3), ce qui justifie d'imposer les mêmes conditions.

que les boucles créées par les points fixes mentionnent toujours une construction de type autre que le récurseur.

Définition 2.1 (Types et pré-types contractifs)**.** *Un type ou pré-type est dit* contractif *si, pour toutes les occurrences de types ou pré-types récursifs de la forme* $\text{REC}\,\mathbf{Y}.\,\mathsf{T}$ *qu'il contient,* T *contient un autre constructeur de type que la variable* $\mathbf{Y}$ *et les récurseurs de la forme* $\text{REC}\,\mathbf{Y}'$*.*

Cependant, la contrainte la plus complexe que l'on imposera sur les pré-types porte sur les types de canaux. En effet, la syntaxe des pré-types de canaux locaux comporte un cas $\text{RW}\langle\mathsf{T}_1, \mathsf{T}_2\rangle$ quels que soient les pré-types T_1 et T_2. Or il est indispensable d'imposer que T_2 soit un sous-type de T_1 pour que les communications se déroulant sur un canal de ce type ne brisent pas les propriétés du typage : dans le cas d'une communication sur un canal de type $\text{RW}\langle\text{RW}\langle\rangle, \text{W}\langle\rangle\rangle$, le récepteur dispose après la communication de plus de droits que l'émetteur. Pour formaliser cette notion de sous-typage et prouver des propriétés à son propos en manipulant les points fixes de façon élégante, commençons par définir la vision coinductive des pré-types où ces points fixes sont « dépliés ».

2.3 Pré-types coinductifs

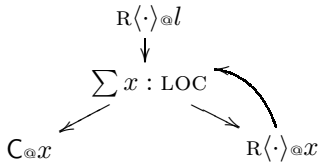
D'habitude, une syntaxe définie par sa grammaire (comme celle donnée figure 2.1) décrit le plus petit ensemble contenant les cas de base (essentiellement le 0-uplet () et la variable $\mathbf{Y}$) et clos par les autres constructions de la syntaxe. Par exemple, si on nomme cet ensemble $\mathcal{P}$, et qu'il contient les types $\mathsf{T}_1, \ldots, \mathsf{T}_n$, la fermeture par la construction de n-uplet impose alors que le type $(\mathsf{T}_1, \ldots, \mathsf{T}_n)$ fasse lui aussi partie de $\mathcal{P}$. Le fait de prendre le plus petit ensemble qui vérifie ces hypothèses implique qu'il ne contient que des types de taille finie, avec d'éventuelles boucles dues à l'opérateur de point fixe $\text{REC}\,\mathbf{Y}.\,\mathsf{T}$.

Il est possible d'adopter, sur les mêmes règles de syntaxe, un point de vue dual. On peut se placer dans l'ensemble des arbres infinis à branchement fini et dont les nœuds sont étiquetés par l'un des cas de la syntaxe et y considérer le plus grand ensemble tel que tous ses éléments respectent la syntaxe. Ainsi le pré-type constitué d'une infinité de $\text{R}\langle\cdot\rangle$ emboîtés, $\mathsf{T} = \text{R}\langle\text{R}\langle\text{R}\langle\cdots\rangle\rangle\rangle$, appartiendra à ce plus grand ensemble car il est de la forme $\text{R}\langle\mathsf{T}'\rangle$ avec $\mathsf{T}' = \mathsf{T}$, donc T' appartient effectivement à l'ensemble. En autorisant les pré-types infinis, on dispose d'une approche alternative à l'opérateur de point fixe $\text{REC}\,\mathbf{Y}.\,\mathsf{T}$ pour autoriser le typage des canaux comme *vois* dans le processus-prospecteur, exactement comme le type $\text{REC}\,\mathbf{Y}.\,\text{R}\langle\mathbf{Y}\rangle$ assure les mêmes propriétés que $\text{R}\langle\text{R}\langle\text{R}\langle\cdots\rangle\rangle\rangle$.

Au lieu de bâtir la notion de sous-typage et toute la théorie qui en découle sur les pré-types inductifs avec l'opérateur de point fixe, on utilisera préférentiellement les pré-types coinductifs sans opérateur de point fixe. On évitera ainsi de faire des choix de représentants de types. Considérons à nouveau l'exemple du canal *vois* du prospecteur. Puisque ce canal est utilisé pour y lire un n-uplet, composé de la localité voisine et d'une paire de canaux qui y sont situés, son type sera simplement une réception d'une somme dépendante. Le type du canal *vois* que l'on pourra utiliser dans la localité l sera ainsi :

$$\text{R}\langle\text{REC}\,\mathbf{Y}.\ \sum x : \text{LOC}.\ (\mathsf{C}_{@}x, \text{R}\langle\mathbf{Y}\rangle_{@}x)\rangle_{@}l$$

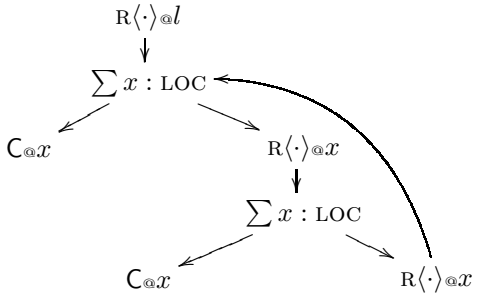
ce que l'on peut représenter de la façon suivante :



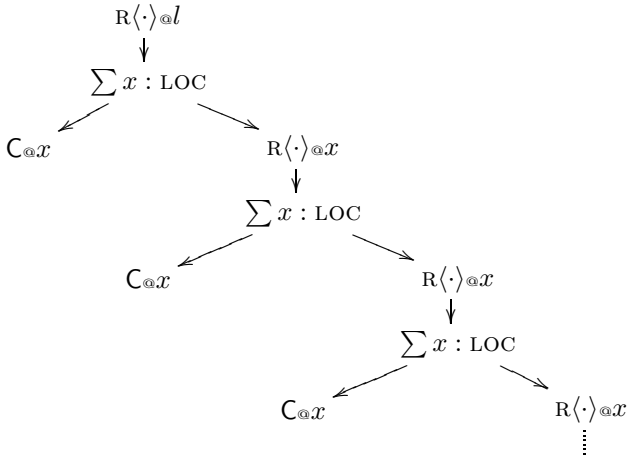
Mais on aurait tout aussi bien pu choisir une autre représentation, comme :

$$\mathrm{R}\langle \mathrm{REC}\,\mathbf{Y}.\ \sum x : \mathrm{LOC}.\ (\mathsf{C}@x, \mathrm{R}\langle \sum x : \mathrm{LOC}.\ (\mathsf{C}@x, \mathrm{R}\langle \mathbf{Y}\rangle@x)\rangle@x)\rangle@l$$

soit



On échappera à ces tergiversations en traduisant les pré-types inductifs mentionnés dans les systèmes et processus par dépliage des points fixes, de sorte à obtenir l'unique[2] arbre correspondant à la fois à tous ces graphes.



Définissons donc la fonction de dépliage $\mathsf{unfold}(\mathsf{T})$ qui associe à tout pré-type inductif son pré-type coinductif. Le pré-type coinductif associé sera donc un arbre fini ou infini dont tous les nœuds et feuilles seront étiquetés par une des constructions de la figure 2.1 mis à part $\mathrm{REC}\,\mathbf{Y}.\ \mathsf{T}$ et $\mathbf{Y}$.

Définition 2.2 (Dépliage)**.** *À tout pré-type* T *contractif,* $\mathsf{unfold}(\mathsf{T})$ *associe l'unique arbre vérifiant les équations suivantes :*

[2]unique à un renommage des variables liées près. Mais on a déjà dit que la liaison était traitée dans les types comme dans les systèmes, et que les α-équivalents sont identifiés.

dépliage des points fixes $\mathit{unfold}(\text{REC}\,\mathbf{Y}.\,\mathsf{T}) = \mathit{unfold}(\mathsf{T}\{^{\text{REC}\,\mathbf{Y}.\,\mathsf{T}}/_{\mathbf{Y}}\})$;

invariance ailleurs *toutes les autres constructions de type sont conservées intactes, par exemple* $\mathit{unfold}(\text{R}\langle\mathsf{T}\rangle) = r\langle\mathit{unfold}(\mathsf{T})\rangle$.

$\mathit{unfold}(\mathsf{T})$ *est le pré-type* coinductif, *ou* arborescent *associé au pré-type inductif* T.

Le pré-type coinductif $\mathsf{unfold}(\mathsf{T})$ n'est défini que quand T est un pré-type contractif. En effet, le cas de dépliage des points fixes permettrait seulement de conclure que $\mathsf{unfold}(\text{REC}\,\mathbf{Y}.\,\mathbf{Y}) = \mathsf{unfold}(\text{REC}\,\mathbf{Y}.\,\mathbf{Y})$, sans plus de précision. Le cas non-contractif est ainsi évité dans cette définition pour la raison qui justifie la définition des pré-types contractifs (définition 2.1) : l'objectif des types et des pré-types est d'identifier un ensemble de capacités liées à une valeur du langage ; le cas dégénéré des pré-types contractifs ne permet plus d'atteindre cet objectif.

Par ailleurs, le fait d'associer un pré-type coinductif à tout pré-type inductif contractif engendre automatiquement une équivalence : deux pré-types inductifs qui partagent le même pré-type coinductif associé seront considérés comme équivalents. Les deux pré-types inductifs envisagés pour le canal *vois* du prospecteur rentrent dans cette catégorie : ces deux types sont interchangeables dans la théorie que l'on élabore ici, dans la mesure où ils permettent de prouver les mêmes propriétés sur les processus où ils apparaissent.

Afin d'éviter toute ambiguïté entre les pré-types inductifs et coinductifs, on les notera sensiblement différemment. Le type inductif

$$\text{R}\langle\sum x : \text{LOC}.\ \mathsf{C}_{@}x\rangle_{@}l$$

sera ainsi écrit

$$\mathsf{r}\langle\sum x : \mathsf{loc}.\ \mathsf{C}_{@}x\rangle_{@}l$$

Nous pouvons maintenant définir la relation de sous-typage sur les pré-types coinductifs.

2.4 Sous-typage

L'intuition sous-jacente à la relation de sous-typage a déjà été exposée. Il s'agit d'une relation d'ordre entre les types, notée $<:$. On dira ainsi $\mathsf{T}_1 <: \mathsf{T}_2$ dès que toute valeur de type T_1 peut être utilisée à la place d'une valeur de type T_2. Par conséquent, on peut considérer que l'ensemble des valeurs de type T_1 est inclus dans celui des valeurs de type T_2. L'ordre de sous-typage reprend donc celui de l'inclusion des ensembles de valeurs. Cela signifie par contre que cet ordre et les inclusions de capacités sont opposés. En effet, revenons sur certains cas « naturels » de sous-typage. Nous avons par exemple déjà croisé plusieurs cas simples : le fait que le type $\mathsf{rw}\langle\mathsf{T}_1, \mathsf{T}_2\rangle$ d'un canal sur lequel on peut aussi bien émettre que recevoir est à la fois un sous-type de $\mathsf{r}\langle\mathsf{T}_1\rangle$ et de $\mathsf{w}\langle\mathsf{T}_2\rangle$, c'est-à-dire les types des canaux sur lesquels une seule des deux capacités reste disponible. Cet « oubli » de capacités correspond donc au passage à un type supérieur dans l'ordre de sous-typage.

Nous avons aussi rencontré le cas où le sous-typage ne porte pas seulement sur le constructeur de tête du type mais aussi sur une sous-partie, en utilisant au type $\mathsf{r}\langle\mathsf{r}\langle\rangle\rangle_{@}l$ un canal de type $\mathsf{rw}\langle\mathsf{rw}\langle\rangle\rangle_{@}l$. On avait proposé de distinguer

Fig. 2.2 Fonction Sub

$$
\begin{aligned}
\mathsf{Sub}(\mathcal{R}) = \quad & \{(\mathsf{r}\langle \mathsf{T}_1\rangle, \mathsf{r}\langle \mathsf{T}_2\rangle) \text{ si } (\mathsf{T}_1, \mathsf{T}_2) \text{ est dans } \mathcal{R}\} \\
\cup \quad & \{(\mathsf{w}\langle \mathsf{T}_1\rangle, \mathsf{w}\langle \mathsf{T}_2\rangle) \text{ si } (\mathsf{T}_2, \mathsf{T}_1) \text{ est dans } \mathcal{R}\} \\
\cup \quad & \{(\mathsf{rw}\langle \mathsf{T}_1^r, \mathsf{T}_1^w\rangle, \mathsf{r}\langle \mathsf{T}_2^r\rangle) \text{ si } (\mathsf{T}_1^w, \mathsf{T}_1^r) \text{ et } (\mathsf{T}_1^r, \mathsf{T}_2^r) \text{ sont dans } \mathcal{R}\} \\
\cup \quad & \{(\mathsf{rw}\langle \mathsf{T}_1^r, \mathsf{T}_1^w\rangle, \mathsf{w}\langle \mathsf{T}_2^w\rangle) \text{ si } (\mathsf{T}_2^w, \mathsf{T}_1^w) \text{ et } (\mathsf{T}_1^w, \mathsf{T}_1^r) \text{ sont dans } \mathcal{R}\} \\
\cup \quad & \{(\mathsf{rw}\langle \mathsf{T}_1^r, \mathsf{T}_1^w\rangle, \mathsf{rw}\langle \mathsf{T}_2^r, \mathsf{T}_2^w\rangle) \text{ si } (\mathsf{T}_2^w, \mathsf{T}_1^w), \\
& \qquad (\mathsf{T}_1^w, \mathsf{T}_1^r) \text{ et } (\mathsf{T}_1^r, \mathsf{T}_2^r) \text{ sont dans } \mathcal{R}\} \\
\cup \quad & \{(\mathsf{loc}, \mathsf{loc})\} \\
\cup \quad & \{((\vec{\mathsf{T}_1}), (\vec{\mathsf{T}_2})) \text{ si } (\mathsf{T}_{1i}, \mathsf{T}_{2i}) \text{ est dans } \mathcal{R} \text{ pour tout } i\} \\
\cup \quad & \{(\textstyle\sum \vec{x} : \vec{\mathsf{loc}}.\ \mathsf{T}_1, \sum \vec{x} : \vec{\mathsf{loc}}.\ \mathsf{T}_2) \text{ si } (\mathsf{T}_1, \mathsf{T}_2) \text{ est dans } \mathcal{R}\} \\
\cup \quad & \{(\mathsf{C}_1 @ x, \mathsf{C}_2 @ x) \text{ si } (\mathsf{C}_1, \mathsf{C}_2) \text{ est dans } \mathcal{R}\}
\end{aligned}
$$

deux étapes pour se convaincre de ce rapport de sous-typage : l'oubli de la capacité d'émission sur le canal lui-même, puis l'oubli des capacités d'émissions de tous les canaux que l'on recevra. De manière plus générale, lorsque T_1 est un sous-type de T_2, $\mathsf{r}\langle \mathsf{T}_1\rangle$ est un sous-type de $\mathsf{r}\langle \mathsf{T}_2\rangle$: sur un canal de type $\mathsf{r}\langle \mathsf{T}_1\rangle$, on reçoit des valeurs de type T_1 c'est-à-dire, en particulier, de type T_2, donc ce canal a aussi le type $\mathsf{r}\langle \mathsf{T}_2\rangle$. On parlera de *covariance* pour les constructions de type qui ne modifient pas l'ordre de sous-typage, comme $\mathsf{r}\langle\cdot\rangle$ qui permet de conclure $\mathsf{r}\langle \mathsf{T}_1\rangle <: \mathsf{r}\langle \mathsf{T}_2\rangle$ sachant $\mathsf{T}_1 <: \mathsf{T}_2$. Les constructions de n-uplets, somme dépendante, localisation d'un type de canal sont également covariantes.

La seule construction de type qui ne soit pas covariante est l'émission $\mathsf{w}\langle\cdot\rangle$. En effet, si $\mathsf{T}_1 <: \mathsf{T}_2$, on peut envoyer des valeurs de type T_1 sur tout canal de type $\mathsf{w}\langle \mathsf{T}_2\rangle$, donc $\mathsf{w}\langle \mathsf{T}_2\rangle$ est un sous-type de $\mathsf{w}\langle \mathsf{T}_1\rangle$. On parlera alors de *contravariance* pour $\mathsf{w}\langle\cdot\rangle$ car $\mathsf{T}_1 <: \mathsf{T}_2$ implique $\mathsf{w}\langle \mathsf{T}_2\rangle <: \mathsf{w}\langle \mathsf{T}_1\rangle$.

Si les pré-types étaient finis, on pourrait définir la relation d'ordre $<:$ par un jeu de règles d'inférence comme la règle ci-dessous. On pourrait alors décider l'ordre entre deux pré-types comparables en fonction de la construction de têtes et de l'ordre établi entre les sous-morceaux du pré-type.

$$\frac{\mathsf{T}_1 <: \mathsf{T}_2}{\mathsf{r}\langle \mathsf{T}_1\rangle <: \mathsf{r}\langle \mathsf{T}_2\rangle}$$

Mais les pré-types sont coinductifs, c'est-à-dire potentiellement infinis. Or les preuves construites par des enchaînements de telles règles d'inférence sont finies, elles ne pourraient donc pas prouver toutes les relations de sous-typage. On formulera par conséquent les inférences possibles sous la forme d'une fonction, Sub, calculant une relation sur les pré-types coinductifs à partir d'une autre relation sur ces pré-types. Si $\mathcal{R}$ est une relation sur les pré-types, $\mathsf{Sub}(\mathcal{R})$ sera la relation contenant toutes les conclusions qu'il est possible de tirer des « hypothèses » contenues dans $\mathcal{R}$. Par exemple, si $\mathcal{R}$ contient le couple $(\mathsf{T}_1, \mathsf{T}_2)$, quel qu'il soit, $\mathsf{Sub}(\mathcal{R})$ contiendra notamment les couples $(\mathsf{r}\langle \mathsf{T}_1\rangle, \mathsf{r}\langle \mathsf{T}_2\rangle)$ et $(\mathsf{w}\langle \mathsf{T}_2\rangle, \mathsf{w}\langle \mathsf{T}_1\rangle)$. La fonction Sub est définie formellement à la figure 2.2.

Bien entendu, l'objectif de cette fonction est de permettre de définir la relation de sous-typage. On l'obtient en constatant que Sub est totale et monotone. Grâce au théorème de Knaster-Tarski, on conclut alors qu'elle admet un plus grand point fixe que l'on notera $\nu\mathsf{Sub}$. Ce plus grand point fixe est la relation recherchée.

Définition 2.3 (Sous-typage)**.** *Le pré-type* T_1 *est un* sous-type *du pré-type* T_2*, ce qui se note* $\mathsf{T}_1 <: \mathsf{T}_2$*, si le couple* $(\mathsf{T}_1, \mathsf{T}_2)$ *apparaît dans la relation* $\nu\mathsf{Sub}$*.*

Cette relation de sous-typage était la dernière notion nécessaire pour pouvoir définir les types.

Définition 2.4 (Types)**.** *Un pré-type coinductif* T *est un type coinductif quand :*
- *pour toutes les occurrences de pré-types de canaux de la forme* $\mathsf{rw}\langle \mathsf{T}^r, \mathsf{T}^w \rangle$ *qu'il contient,* $\mathsf{T}^w <: \mathsf{T}^r$ *;*
- *pour toutes les occurrences de pré-types de la forme* C *qu'il contient,* C *est clos.*

Un pré-type inductif T *est un type inductif quand :*
- *il est contractif ;*
- *toutes les variables de type récursif qu'il contient sont liées ;*
- $\mathsf{unfold}(\mathsf{T})$ *est un type coinductif.*

2.5 Théorie des types coinductifs, théorie coinductive des types

Définir le sous-typage par un plus grand point fixe masque le fait qu'il s'agit d'un pré-ordre. Nous allons donc nous attacher ici à prouver cette propriété du sous-typage et à caractériser l'équivalence engendrée par le pré-ordre : nous pourrons alors considérer les types coinductifs à équivalence près. Sans ces propriétés, la structure de l'ensemble de types ne pourrait correspondre à l'objectif fixé. En effet, l'inégalité $\mathsf{T}_1 <: \mathsf{T}_2$ doit garantir que les valeurs de type T_1 sont aussi de type T_2, donc deux types équivalents correspondent naturellement aux mêmes valeurs et la relation doit évidemment être réflexive et transitive.

Nous nous intéresserons ensuite aux bornes inférieures et supérieures engendrées par l'ordre. Puisque l'on associe un type à chaque nom et chaque variable, la question de fond sous-jacente sera de savoir quand un identifiant peut avoir deux types distincts donnés : nous chercherons donc à caractériser quand des types admettent une borne inférieure. Cette caractérisation prendra tout son sens lorsque, au cours d'un calcul, un processus apprendra un même nom à deux types différents : on voudra dans ce cas-là permettre au processus d'utiliser pleinement ce nom, en combinant les différentes capacités qu'il possède sur ce nom, la borne inférieure correspondant justement à cette combinaison des capacités.

Pour cela nous donnerons donc une relation portant sur les types coinductifs et nous prouverons que cette relation définit effectivement les bornes inférieures et supérieures. Puis nous prouverons que deux types ayant un sous-type commun possèdent une borne inférieure. Enfin, nous prouverons que la borne inférieure est associative ce qui, couplé avec sa commutativité, nous autorisera par la suite à considérer la borne inférieure d'ensemble de types.

Commençons donc par prouver que $<:$ est un pré-ordre. La preuve de cette propriété suivra une démarche habituelle pour la coinduction reposant sur les post-points fixes.

Définition 2.5 (Post-point fixe d'une fonction)**.** *Un élément* x *est un* post-point fixe *d'une fonction* f *dès que* $x \leq f(x)$*.*

Or le plus grand point fixe d'une fonction portant sur les relations est aussi son plus grand post-point fixe. Ainsi le sous-typage est le plus grand post-point fixe de Sub. Pour prouver que T_1 est un sous-type de T_2, on procédera donc souvent en deux étapes :

- trouver une relation $\mathcal{S}$ telle que $(\mathsf{T}_1, \mathsf{T}_2)$ soit dans cette relation ;
- prouver que cette relation $\mathcal{S}$ est un post-point fixe de Sub.

La seconde partie de la preuve impliquera que la relation $\mathcal{S}$ est incluse dans $<:$ puisqu'il s'agit de la plus grande relation vérifiant cette propriété, alors que la première partie assurera qu'elle contient notamment le couple $(\mathsf{T}_1, \mathsf{T}_2)$. Cette démarche pourra aussi être utilisée pour toutes les relations définies comme plus grands points fixes d'une fonction.

On pourra aussi utiliser une version un peu plus faible de la condition de post-point fixe dans le cas particulier de la fonction Sub. En effet, Sub vérifie la propriété suivante :

Proposition 2.6. *Pour une relation $\mathcal{R}$, si $\mathcal{R} \subseteq \mathsf{Sub}(\mathcal{R}^+)$ alors $\mathcal{R}^+ \subseteq \mathsf{Sub}(\mathcal{R}^+)$.*

Démonstration. La preuve procède simplement en remarquant que si $\mathcal{R}_1$ et $\mathcal{R}_2$ sont deux relations contenues dans $\mathsf{Sub}(\mathcal{R}^+)$ alors leur composition l'est aussi. Considérons ainsi un couple $(\mathsf{T}_1, \mathsf{T}_3)$ de $\mathcal{R}_1 \circ \mathcal{R}_2$ et T_2 qui le prouve c'est-à-dire tel que $(\mathsf{T}_1, \mathsf{T}_2) \in \mathcal{R}_1$ et $(\mathsf{T}_2, \mathsf{T}_3) \in \mathcal{R}_2$. Étudions deux exemples de cas possibles pour le couple $(\mathsf{T}_1, \mathsf{T}_2)$ qui est inclus dans $\mathsf{Sub}(\mathcal{R}^+)$.

- S'il s'agit d'un couple $(\mathsf{r}\langle \mathsf{T}'_1\rangle, \mathsf{r}\langle \mathsf{T}'_2\rangle)$, le fait que $(\mathsf{T}_2, \mathsf{T}_3)$ soit aussi dans $\mathsf{Sub}(\mathcal{R}^+)$ impose alors que T_3 soit de la forme $\mathsf{r}\langle \mathsf{T}'_3\rangle$. Puisque les couples $(\mathsf{T}'_1, \mathsf{T}'_2)$ et $(\mathsf{T}'_2, \mathsf{T}'_3)$ sont dans $\mathcal{R}^+$ et que cette relation est naturellement transitive, $(\mathsf{T}'_1, \mathsf{T}'_3)$ doit aussi y être, ce qui implique $(\mathsf{T}_1, \mathsf{T}_3) \in \mathsf{Sub}(\mathcal{R}^+)$.
- S'il s'agit d'un couple $(\mathsf{rw}\langle \mathsf{T}^r_1, \mathsf{T}^w_1\rangle, \mathsf{rw}\langle \mathsf{T}^r_2, \mathsf{T}^w_2\rangle)$, T_3 peut prendre les formes $\mathsf{r}\langle \mathsf{T}^r_3\rangle$, $\mathsf{w}\langle \mathsf{T}^w_3\rangle$ ou $\mathsf{rw}\langle \mathsf{T}^r_3, \mathsf{T}^w_3\rangle$). Dans tous les cas, les trois couples $(\mathsf{T}^w_2, \mathsf{T}^w_1)$, $(\mathsf{T}^w_1, \mathsf{T}^r_1)$, $(\mathsf{T}^r_1, \mathsf{T}^r_2)$ sont dans $\mathcal{R}^+$. Or $(\mathsf{T}_2, \mathsf{T}_3)$ permet de déduire que $(\mathsf{T}^r_2, \mathsf{T}^r_3)$ ou $(\mathsf{T}^w_3, \mathsf{T}^w_2)$ ou les deux sont dans $\mathcal{R}^+$. Donc les couples correspondants $(\mathsf{T}^r_1, \mathsf{T}^r_3)$ ou $(\mathsf{T}^w_3, \mathsf{T}^w_1)$ ou les deux y sont aussi et impliquent que $(\mathsf{T}_1, \mathsf{T}_3)$ soit dans $\mathsf{Sub}(\mathcal{R}^+)$.

Les autres cas sont semblables.

Cette propriété permet de prouver par une induction simple que $\mathcal{R}^n$ est incluse dans $\mathsf{Sub}(\mathcal{R}^+)$, donc $\mathcal{R}^+$ l'est aussi. □

Prouvons donc $<:$ est un pré-ordre.

Proposition 2.7. *$<:$ est un pré-ordre partiel sur les types coinductifs.*

Démonstration. Prouvons que $<:$ est réflexif et transitif.

Pour montrer que $<:$ est réflexif, il suffit de prouver que la relation

$$\mathcal{R} = \{(\mathsf{T}, \mathsf{T}) \mid \mathsf{T} \text{ un type quelconque}\} \cup \nu\mathsf{Sub}$$

est un post-point fixe de Sub. On obtient facilement ce résultat en prenant un couple $(\mathsf{T}_1, \mathsf{T}_2)$ dans $\mathcal{R}$. Si ce couple provient de la partie $\nu\mathsf{Sub}$ de $\mathcal{R}$, son appartenance à $\mathsf{Sub}(\mathcal{R})$ est immédiate car il est dans la relation $\mathsf{Sub}(\nu\mathsf{Sub})$ et Sub est monotone. Si $(\mathsf{T}_1, \mathsf{T}_2)$ est de la forme (T, T), on raisonne sur la forme du type T. Considérons simplement ici le cas $\mathsf{T} = \mathsf{rw}\langle \mathsf{T}^r, \mathsf{T}^w\rangle$. On voit immédiatement que $\mathsf{T}^r \ \mathcal{R} \ \mathsf{T}^r$ et $\mathsf{T}^w \ \mathcal{R} \ \mathsf{T}^w$. $\mathsf{T}^w \ \mathcal{R} \ \mathsf{T}^r$ découle du fait que $\mathcal{R}$ contient $\nu\mathsf{Sub}$ et que la

bonne formation du type T impose ce sous-typage à toutes les occurrences de $\mathsf{rw}\langle \mathsf{T}_0^r, \mathsf{T}_0^w\rangle$ dans T. Les autres cas sont similaires.

Pour prouver que $<:$ est transitive, montrons que $\nu\mathsf{Sub} \cup \nu\mathsf{Sub} \circ \nu\mathsf{Sub}$ est un post-point fixe de Sub. Soit un couple $(\mathsf{T}_1, \mathsf{T}_3)$ de $\nu\mathsf{Sub} \cup \nu\mathsf{Sub} \circ \nu\mathsf{Sub}$. S'il est dans $\nu\mathsf{Sub}$, le résultat est immédiat. Sinon, il existe un T_2 tel que $\mathsf{T}_1 <: \mathsf{T}_2$ et $\mathsf{T}_2 <: \mathsf{T}_3$. Raisonnons alors en cascade suivant les formes de T_1, T_2 et T_3, chacune contraignant les suivantes. Considérons par exemple le cas où les types sont $\mathsf{rw}\langle \mathsf{T}_1^r, \mathsf{T}_1^w\rangle$, $\mathsf{rw}\langle \mathsf{T}_2^r, \mathsf{T}_2^w\rangle$ et $\mathsf{r}\langle \mathsf{T}_3^r\rangle$. On peut alors en déduire que $\mathsf{T}_1^w <: \mathsf{T}_1^r$ et $\mathsf{T}_1^r <: \mathsf{T}_2^r <: \mathsf{T}_3^r$, c'est-à-dire que $(\mathsf{T}_1^w, \mathsf{T}_1^r)$ et $(\mathsf{T}_1^r, \mathsf{T}_3^r)$ sont dans $\nu\mathsf{Sub} \cup \nu\mathsf{Sub} \circ \nu\mathsf{Sub}$. D'où l'on conclut que $(\mathsf{T}_1, \mathsf{T}_3)$ est dans $\mathsf{Sub}(\nu\mathsf{Sub} \cup \nu\mathsf{Sub} \circ \nu\mathsf{Sub})$. Les autres cas sont là aussi similaires. □

Comme tout pré-ordre, $<:$ engendre une équivalence sur les types, en considérant que deux types T_1 et T_2 sont équivalents dès lors que $\mathsf{T}_1 <: \mathsf{T}_2$ et $\mathsf{T}_2 <: \mathsf{T}_1$. Cette équivalence coïncide avec l'égalité extensionnelle selon laquelle deux types sont égaux à la condition que leurs constructeurs de type de tête soient égaux et que les sous-parties soient égales. Cette égalité peut alors se définir comme le plus grand point fixe d'une fonction sur les relations entre types.

$$
\begin{aligned}
\mathsf{Eq}(\mathcal{R}) = \quad & \{(\mathsf{r}\langle \mathsf{T}_1\rangle, \mathsf{r}\langle \mathsf{T}_2\rangle) \text{ si } (\mathsf{T}_1, \mathsf{T}_2) \text{ est dans } \mathcal{R}\} \\
\cup \quad & \{(\mathsf{w}\langle \mathsf{T}_1\rangle, \mathsf{w}\langle \mathsf{T}_2\rangle) \text{ si } (\mathsf{T}_1, \mathsf{T}_2) \text{ est dans } \mathcal{R}\} \\
\cup \quad & \{(\mathsf{rw}\langle \mathsf{T}_1^r, \mathsf{T}_1^w\rangle, \mathsf{rw}\langle \mathsf{T}_2^r, \mathsf{T}_2^w\rangle) \text{ si } (\mathsf{T}_1^w, \mathsf{T}_2^w) \text{ et } (\mathsf{T}_1^r, \mathsf{T}_2^r) \text{ sont dans } \mathcal{R}\} \\
\cup \quad & \{(\mathsf{loc}, \mathsf{loc})\} \\
\cup \quad & \{((\vec{\mathsf{T}_1}), (\vec{\mathsf{T}_2})) \text{ si } (\mathsf{T}_{1i}, \mathsf{T}_{2i}) \text{ est dans } \mathcal{R} \text{ pour tout } i\} \\
\cup \quad & \{(\textstyle\sum \vec{x} : \vec{\mathsf{loc}}.\ \mathsf{T}_1, \sum \vec{x} : \vec{\mathsf{loc}}.\ \mathsf{T}_2) \text{ si } (\mathsf{T}_1, \mathsf{T}_2) \text{ est dans } \mathcal{R}\} \\
\cup \quad & \{(\mathsf{C}_{1@}x, \mathsf{C}_{2@}x) \text{ si } (\mathsf{C}_1, \mathsf{C}_2) \text{ est dans } \mathcal{R}\}
\end{aligned}
$$

Montrons que cette égalité est effectivement l'équivalence engendrée par le sous-typage.

Proposition 2.8. *$\nu\mathsf{Eq}$ est l'équivalence engendrée par le pré-ordre $<:$, c'est-à-dire que $(\mathsf{T}_1, \mathsf{T}_2)$ est dans $\nu\mathsf{Eq}$ si et seulement si $\mathsf{T}_1 <: \mathsf{T}_2$ et $\mathsf{T}_2 <: \mathsf{T}_1$.*

Démonstration. Remarquons tout d'abord que prouver « $(\mathsf{T}_1, \mathsf{T}_2) \in \nu\mathsf{Eq}$ si et seulement si $\mathsf{T}_1 <: \mathsf{T}_2$ et $\mathsf{T}_2 <: \mathsf{T}_1$ » impliquera directement le fait qu'il s'agisse d'une équivalence. En effet, les réflexivité et transitivité de $<:$ entraîneront directement celles de $\nu\mathsf{Eq}$, et la symétrie découlera de la condition symétrique « $\mathsf{T}_1 <: \mathsf{T}_2$ et $\mathsf{T}_2 <: \mathsf{T}_1$ ».

Commençons par prouver que $\mathsf{T}_1 <: \mathsf{T}_2$ et $\mathsf{T}_2 <: \mathsf{T}_1$ implique $(\mathsf{T}_1, \mathsf{T}_2) \in \nu\mathsf{Eq}$. Pour cela, il suffit de prouver que la relation

$$\mathcal{R} = \{(\mathsf{T}_1, \mathsf{T}_2) \mid \mathsf{T}_1 <: \mathsf{T}_2, \mathsf{T}_2 <: \mathsf{T}_1\}$$

est un post-point fixe de Eq. Soit $(\mathsf{T}_1, \mathsf{T}_2)$ un couple de $\mathcal{R}$. $(\mathsf{T}_1, \mathsf{T}_2) \in \mathsf{Sub}(\nu\mathsf{Sub})$ donc il doit être dans l'un des cas de la définition de Sub. L'hypothèse que $(\mathsf{T}_2, \mathsf{T}_1)$ est aussi dans $\mathsf{Sub}(\nu\mathsf{Sub})$ interdit qu'il s'agisse de l'un des cas asymétriques $(\mathsf{rw}\langle \mathsf{T}_1^r, \mathsf{T}_1^w\rangle, \mathsf{r}\langle \mathsf{T}_2^r\rangle)$ ou $(\mathsf{rw}\langle \mathsf{T}_1^r, \mathsf{T}_1^w\rangle, \mathsf{w}\langle \mathsf{T}_2^w\rangle)$. Supposons par exemple qu'il s'agisse de $(\mathsf{rw}\langle \mathsf{T}_1^r, \mathsf{T}_1^w\rangle, \mathsf{rw}\langle \mathsf{T}_2^r, \mathsf{T}_2^w\rangle)$. Nous pouvons alors en déduire en particulier que $\mathsf{T}_1^w <: \mathsf{T}_2^w$, $\mathsf{T}_2^w <: \mathsf{T}_1^w$, $\mathsf{T}_1^r <: \mathsf{T}_2^r$ et $\mathsf{T}_2^r <: \mathsf{T}_1^r$ c'est-à-dire que les couples $(\mathsf{T}_1^r, \mathsf{T}_2^r)$ et $(\mathsf{T}_1^w, \mathsf{T}_2^w)$ sont dans $\mathcal{R}$. $(\mathsf{T}_1, \mathsf{T}_2)$ est donc dans $\mathsf{Eq}(\mathcal{R})$. Les autres cas sont similaires.

Pour la réciproque, il suffit de montrer que $\nu\mathsf{Eq} \cup \nu\mathsf{Eq}^{-1}$ est un post-point fixe de Sub. Soit $(\mathsf{T}_1, \mathsf{T}_2)$ un couple de $\nu\mathsf{Eq}$. $\nu\mathsf{Eq} = \mathsf{Eq}(\nu\mathsf{Eq})$ implique que le

couple $(\mathsf{T}_1, \mathsf{T}_2)$ est de la forme d'un des cas de la définition de Eq. Supposons par exemple qu'il soit de la forme $(\mathsf{r}\langle \mathsf{T}'_1\rangle, \mathsf{r}\langle \mathsf{T}'_2\rangle)$, ce qui implique que $(\mathsf{T}'_1, \mathsf{T}'_2)$ est aussi dans $\nu\mathsf{Eq}$. Nous pouvons alors conclure directement que $(\mathsf{T}_1, \mathsf{T}_2)$ est aussi dans $\mathsf{Sub}(\nu\mathsf{Eq}) \subseteq \mathsf{Sub}(\nu\mathsf{Eq} \cup \nu\mathsf{Eq}^{-1})$. Les autres cas, qu'ils soient dans $\nu\mathsf{Eq}$ ou dans $\nu\mathsf{Eq}^{-1}$, sont similaires, en utilisant la symétrie de $\nu\mathsf{Eq} \cup \nu\mathsf{Eq}^{-1}$ pour les constructions contravariantes. □

L'ensemble des types que l'on utilisera dans la suite sera quotienté par cette équivalence $\nu\mathsf{Eq}$, ce qui permettra d'utiliser la méthode décrite au début de cette section pour prouver l'égalité de deux types. On écrira donc $\mathsf{T}_1 = \mathsf{T}_2$ dès que le couple $(\mathsf{T}_1, \mathsf{T}_2)$ est dans $\nu\mathsf{Eq}$. Le choix de cette notion d'égalité entraîne par ailleurs le fait que $<:$ est un ordre partiel sur l'ensemble des types.

Comme annoncé au début de cette section, regardons maintenant les bornes inférieures et supérieures engendrées par cet ordre partiel. Dans le cadre du calcul, la question de l'existence et de la valeur de ces bornes correspond à identifier quel est le type auquel un nom est connu par un processus qui acquiert successivement des connaissances sur ce nom, en le recevant à des types différents. De la même façon que la fonction Eq permet de préciser et de manipuler l'équivalence engendrée par la relation de sous-typage sur les types coinductifs, définissons une fonction dont le plus grand point fixe coïncidera avec les bornes. Pour définir les bornes, il faudra que la relation engendrée par cette fonction ne porte pas sur des paires mais sur des triplets de la forme ${}_{\sqcap}(\mathsf{T}_1, \mathsf{T}_2, \mathsf{T}_3)$, signifiant que T_3 est la borne inférieure de T_1 et T_2. Cependant, la contravariance du constructeur de type $\mathsf{w}\langle\cdot\rangle$ impose que la borne inférieure des types $\mathsf{w}\langle \mathsf{T}_1\rangle$ et $\mathsf{w}\langle \mathsf{T}_2\rangle$ soit un type $\mathsf{w}\langle \mathsf{T}_3\rangle$ où T_3 est la borne supérieure des types T_1 et T_2. Cette relation ne pourra ainsi pas se contenter de parler des triplets ${}_{\sqcap}(\mathsf{T}_1, \mathsf{T}_2, \mathsf{T}_3)$ mais devra aussi contenir des triplets ${}_{\sqcup}(\mathsf{T}_1, \mathsf{T}_2, \mathsf{T}_3)$ pour indiquer que T_3 est la borne supérieure de T_1 et T_2.

Or le calcul des bornes de deux types peut s'avérer problématique. Bien entendu, les bornes n'existent pas pour tous les types, par exemple un type de canal comme $\mathsf{r}\langle\rangle_{@l}$ et un type de localité loc sont totalement inconciliables. Mais il est possible d'avoir des conflits plus subtils. Prenons l'exemple de la borne supérieure des deux types $\mathsf{rw}\langle\mathsf{rw}\langle\mathsf{r}\langle\rangle\rangle\rangle$ et $\mathsf{rw}\langle\mathsf{r}\langle\mathsf{rw}\langle\rangle\rangle, \mathsf{rw}\langle\mathsf{rw}\langle\rangle\rangle\rangle$. Suivant la décomposition imposée par l'ordre de sous-typage, l'intuition voudrait que leur borne supérieure soit un type de canal sur lequel les valeurs sont reçues à la borne supérieure des types $\mathsf{rw}\langle\mathsf{r}\langle\rangle\rangle$ et $\mathsf{r}\langle\mathsf{rw}\langle\rangle\rangle$ et émises à la borne inférieure de $\mathsf{rw}\langle\mathsf{r}\langle\rangle\rangle$ et $\mathsf{rw}\langle\mathsf{rw}\langle\rangle\rangle$. On peut aisément se convaincre que la borne supérieure des deux premiers doit être $\mathsf{r}\langle\mathsf{r}\langle\rangle\rangle$. Par contre, si elle existait, la borne inférieure des deux seconds devrait être un type de canal sur lequel recevoir des valeurs de type $\mathsf{rw}\langle\rangle$ et émettre des valeurs de type $\mathsf{r}\langle\rangle$. Or ce type ne vérifie pas la principale contrainte sur les types de canaux disant que le type en émission doit être un sous-type de celui en réception, sans quoi le canal permettrait de recevoir plus de capacités que n'en sont émises. Lorsqu'il est ainsi impossible de trouver une borne inférieure pour les deux types en émission, la borne supérieure de deux types de canaux n'autorisera que des réceptions.

Afin de donner un critère simple pour déterminer si la borne supérieure de deux types de la forme $\mathsf{rw}\langle\cdot\rangle$ autorisera à effectuer des émissions et des réceptions, on a recours à une notion de compatibilité.

Fig. 2.3 Fonction MeetJoin

$$
\begin{array}{ll}
\mathsf{MeetJoin}(\mathcal{R}) = & \\
 & \{\sqcap(\mathsf{r}\langle \mathsf{T}_1^r\rangle, \mathsf{r}\langle \mathsf{T}_2^r\rangle, \mathsf{r}\langle \mathsf{T}_3^r\rangle) \text{ si } \sqcap(\mathsf{T}_1^r, \mathsf{T}_2^r, \mathsf{T}_3^r) \text{ est dans } \mathcal{R}\} \\
\cup & \{\sqcap(\mathsf{w}\langle \mathsf{T}_1^w\rangle, \mathsf{w}\langle \mathsf{T}_2^w\rangle, \mathsf{w}\langle \mathsf{T}_3^w\rangle) \text{ si } \sqcup(\mathsf{T}_1^w, \mathsf{T}_2^w, \mathsf{T}_3^w) \text{ est dans } \mathcal{R}\} \\
\cup & \{\sqcap(\mathsf{r}\langle \mathsf{T}_1^r\rangle, \mathsf{w}\langle \mathsf{T}_2^w\rangle, \mathsf{rw}\langle \mathsf{T}_1^r, \mathsf{T}_2^w\rangle) \text{ si } \mathsf{T}_2^w <: \mathsf{T}_1^r\} \\
\cup & \{\sqcap(\mathsf{w}\langle \mathsf{T}_1^w\rangle, \mathsf{r}\langle \mathsf{T}_2^r\rangle, \mathsf{rw}\langle \mathsf{T}_2^r, \mathsf{T}_1^w\rangle) \text{ si } \mathsf{T}_1^w <: \mathsf{T}_2^r\} \\
\cup & \{\sqcap(\mathsf{rw}\langle \mathsf{T}_1^r, \mathsf{T}_1^w\rangle, \mathsf{r}\langle \mathsf{T}_2^r\rangle, \mathsf{rw}\langle \mathsf{T}_3^r, \mathsf{T}_1^w\rangle) \text{ si } \sqcap(\mathsf{T}_1^r, \mathsf{T}_2^r, \mathsf{T}_3^r) \text{ est dans } \mathcal{R} \text{ et } \mathsf{T}_1^w <: \mathsf{T}_3^r\} \\
\cup & \{\sqcap(\mathsf{rw}\langle \mathsf{T}_1^r, \mathsf{T}_1^w\rangle, \mathsf{w}\langle \mathsf{T}_2^w\rangle, \mathsf{rw}\langle \mathsf{T}_1^r, \mathsf{T}_3^w\rangle) \text{ si } \sqcup(\mathsf{T}_1^w, \mathsf{T}_2^w, \mathsf{T}_3^w) \text{ est dans } \mathcal{R} \text{ et } \mathsf{T}_3^w <: \mathsf{T}_1^r\} \\
\cup & \{\sqcap(\mathsf{r}\langle \mathsf{T}_1^r\rangle, \mathsf{rw}\langle \mathsf{T}_2^r, \mathsf{T}_2^w\rangle, \mathsf{rw}\langle \mathsf{T}_3^r, \mathsf{T}_2^w\rangle) \text{ si } \sqcap(\mathsf{T}_1^r, \mathsf{T}_2^r, \mathsf{T}_3^r) \text{ est dans } \mathcal{R} \text{ et } \mathsf{T}_2^w <: \mathsf{T}_3^r\} \\
\cup & \{\sqcap(\mathsf{w}\langle \mathsf{T}_1^w\rangle, \mathsf{rw}\langle \mathsf{T}_2^r, \mathsf{T}_2^w\rangle, \mathsf{rw}\langle \mathsf{T}_2^r, \mathsf{T}_3^w\rangle) \text{ si } \sqcup(\mathsf{T}_1^w, \mathsf{T}_2^w, \mathsf{T}_3^w) \text{ est dans } \mathcal{R} \text{ et } \mathsf{T}_3^w <: \mathsf{T}_2^r\} \\
\cup & \{\sqcap(\mathsf{rw}\langle \mathsf{T}_1^r, \mathsf{T}_1^w\rangle, \mathsf{rw}\langle \mathsf{T}_2^r, \mathsf{T}_2^w\rangle, \mathsf{rw}\langle \mathsf{T}_3^r, \mathsf{T}_3^w\rangle) \\
 & \quad \text{si } \sqcup(\mathsf{T}_1^w, \mathsf{T}_2^w, \mathsf{T}_3^w), \sqcap(\mathsf{T}_1^r, \mathsf{T}_2^r, \mathsf{T}_3^r) \text{ sont dans } \mathcal{R} \text{ et } \mathsf{T}_3^w <: \mathsf{T}_3^r\} \\
\cup & \{\sqcap(\mathsf{loc}, \mathsf{loc}, \mathsf{loc})\} \\
\cup & \{\sqcap((\vec{\mathsf{T}_1}), (\vec{\mathsf{T}_2}), (\vec{\mathsf{T}_3})) \text{ si } \sqcap(\mathsf{T}_{1i}, \mathsf{T}_{2i}, \mathsf{T}_{3i}) \text{ est dans } \mathcal{R} \text{ pour tout } i\} \\
\cup & \{\sqcap(\sum \vec{x} : \vec{\mathsf{loc}}.\ \mathsf{T}_1, \sum \vec{x} : \vec{\mathsf{loc}}.\ \mathsf{T}_2, \sum \vec{x} : \vec{\mathsf{loc}}.\ \mathsf{T}_3) \text{ si } \sqcap(\mathsf{T}_1, \mathsf{T}_2, \mathsf{T}_3) \text{ est dans } \mathcal{R}\} \\
\cup & \{\sqcap(\mathsf{C}_1{}_{@s}, \mathsf{C}_2{}_{@s}, \mathsf{C}_3{}_{@s}) \text{ si } \sqcap(\mathsf{C}_1, \mathsf{C}_2, \mathsf{C}_3) \text{ est dans } \mathcal{R}\} \\
\cup & \{\sqcup(\mathsf{r}\langle \mathsf{T}_1^r\rangle, \mathsf{r}\langle \mathsf{T}_2^r\rangle, \mathsf{r}\langle \mathsf{T}_3^r\rangle) \text{ si } \sqcup(\mathsf{T}_1^r, \mathsf{T}_2^r, \mathsf{T}_3^r) \text{ est dans } \mathcal{R}\} \\
\cup & \{\sqcup(\mathsf{w}\langle \mathsf{T}_1^w\rangle, \mathsf{w}\langle \mathsf{T}_2^w\rangle, \mathsf{w}\langle \mathsf{T}_3^w\rangle) \text{ si } \sqcap(\mathsf{T}_1^w, \mathsf{T}_2^w, \mathsf{T}_3^w) \text{ est dans } \mathcal{R}\} \\
\cup & \{\sqcup(\mathsf{rw}\langle \mathsf{T}_1^r, \mathsf{T}_1^w\rangle, \mathsf{r}\langle \mathsf{T}_2^r\rangle, \mathsf{r}\langle \mathsf{T}_3^r\rangle) \text{ si } \sqcup(\mathsf{T}_1^r, \mathsf{T}_2^r, \mathsf{T}_3^r) \text{ est dans } \mathcal{R} \text{ et } \mathsf{T}_1^w <: \mathsf{T}_1^r\} \\
\cup & \{\sqcup(\mathsf{rw}\langle \mathsf{T}_1^r, \mathsf{T}_1^w\rangle, \mathsf{w}\langle \mathsf{T}_2^w\rangle, \mathsf{w}\langle \mathsf{T}_3^w\rangle) \text{ si } \sqcap(\mathsf{T}_1^w, \mathsf{T}_2^w, \mathsf{T}_3^w) \text{ est dans } \mathcal{R} \text{ et } \mathsf{T}_1^w <: \mathsf{T}_1^r\} \\
\cup & \{\sqcup(\mathsf{r}\langle \mathsf{T}_1^r\rangle, \mathsf{rw}\langle \mathsf{T}_2^r, \mathsf{T}_2^w\rangle, \mathsf{r}\langle \mathsf{T}_3^r\rangle) \text{ si } \sqcup(\mathsf{T}_1^r, \mathsf{T}_2^r, \mathsf{T}_3^r) \text{ est dans } \mathcal{R} \text{ et } \mathsf{T}_2^w <: \mathsf{T}_2^r\} \\
\cup & \{\sqcup(\mathsf{w}\langle \mathsf{T}_1^w\rangle, \mathsf{rw}\langle \mathsf{T}_2^r, \mathsf{T}_2^w\rangle, \mathsf{w}\langle \mathsf{T}_3^w\rangle) \text{ si } \sqcap(\mathsf{T}_1^w, \mathsf{T}_2^w, \mathsf{T}_3^w) \text{ est dans } \mathcal{R} \text{ et } \mathsf{T}_2^w <: \mathsf{T}_2^r\} \\
\cup & \{\sqcup(\mathsf{rw}\langle \mathsf{T}_1^r, \mathsf{T}_1^w\rangle, \mathsf{rw}\langle \mathsf{T}_2^r, \mathsf{T}_2^w\rangle, \mathsf{r}\langle \mathsf{T}_3^r\rangle) \\
 & \quad \text{si } \mathsf{T}_1^r \uparrow \mathsf{T}_2^r, \sqcup(\mathsf{T}_1^r, \mathsf{T}_2^r, \mathsf{T}_3^r) \text{ est dans } \mathcal{R}, \mathsf{T}_1^w \not\downarrow \mathsf{T}_2^w, \mathsf{T}_1^w <: \mathsf{T}_1^r \text{ et } \mathsf{T}_2^w <: \mathsf{T}_2^r\} \\
\cup & \{\sqcup(\mathsf{rw}\langle \mathsf{T}_1^r, \mathsf{T}_1^w\rangle, \mathsf{rw}\langle \mathsf{T}_2^r, \mathsf{T}_2^w\rangle, \mathsf{w}\langle \mathsf{T}_3^w\rangle) \\
 & \quad \text{si } \mathsf{T}_1^w \downarrow \mathsf{T}_2^w, \sqcap(\mathsf{T}_1^w, \mathsf{T}_2^w, \mathsf{T}_3^w) \text{ est dans } \mathcal{R}, \mathsf{T}_1^r \not\uparrow \mathsf{T}_2^r, \mathsf{T}_1^w <: \mathsf{T}_1^r \text{ et } \mathsf{T}_2^w <: \mathsf{T}_2^r\} \\
\cup & \{\sqcup(\mathsf{rw}\langle \mathsf{T}_1^r, \mathsf{T}_1^w\rangle, \mathsf{rw}\langle \mathsf{T}_2^r, \mathsf{T}_2^w\rangle, \mathsf{rw}\langle \mathsf{T}_3^r, \mathsf{T}_3^w\rangle) \\
 & \quad \text{si } \mathsf{T}_1^w \downarrow \mathsf{T}_2^w, \mathsf{T}_1^r \uparrow \mathsf{T}_2^r, \sqcap(\mathsf{T}_1^w, \mathsf{T}_2^w, \mathsf{T}_3^w), \sqcup(\mathsf{T}_1^r, \mathsf{T}_2^r, \mathsf{T}_3^r) \text{ sont dans } \mathcal{R}, \\
 & \quad\ \mathsf{T}_1^w <: \mathsf{T}_1^r \text{ et } \mathsf{T}_2^w <: \mathsf{T}_2^r\} \\
\cup & \{\sqcup(\mathsf{loc}, \mathsf{loc}, \mathsf{loc})\} \\
\cup & \{\sqcup((\vec{\mathsf{T}_1}), (\vec{\mathsf{T}_2}), (\vec{\mathsf{T}_3})) \text{ si } \sqcup(\mathsf{T}_{1i}, \mathsf{T}_{2i}, \mathsf{T}_{3i}) \text{ est dans } \mathcal{R} \text{ pour tout } i\} \\
\cup & \{\sqcup(\sum \vec{x} : \vec{\mathsf{loc}}.\ \mathsf{T}_1, \sum \vec{x} : \vec{\mathsf{loc}}.\ \mathsf{T}_2, \sum \vec{x} : \vec{\mathsf{loc}}.\ \mathsf{T}_3) \text{ si } \sqcup(\mathsf{T}_1, \mathsf{T}_2, \mathsf{T}_3) \text{ est dans } \mathcal{R}\} \\
\cup & \{\sqcup(\mathsf{C}_1{}_{@s}, \mathsf{C}_2{}_{@s}, \mathsf{C}_3{}_{@s}) \text{ si } \sqcup(\mathsf{C}_1, \mathsf{C}_2, \mathsf{C}_3) \text{ est dans } \mathcal{R}\}
\end{array}
$$

Définition 2.9 (Compatibilité de types)**.** *On dit que les types* T_1 *et* T_2 *sont* $\downarrow$-compatibles *ou simplement* compatibles*, ce que l'on note* $\mathsf{T}_1 \downarrow \mathsf{T}_2$*, quand ils admettent un sous-type commun.*

Dualement, on dit qu'ils sont $\uparrow$-compatibles*, ce que l'on note* $\mathsf{T}_1 \uparrow \mathsf{T}_2$*, quand ils admettent un sur-type commun.*

La fonction MeetJoin conçue pour obtenir une définition manipulable des bornes est donnée figure 2.3. Tout comme les fonctions Sub et Eq, elle est monotone et totale, elle possède donc un plus grand point fixe. On notera ainsi $\mathsf{T}_1 \sqcap \mathsf{T}_2 = \mathsf{T}_3$ quand le triplet $\sqcap(\mathsf{T}_1, \mathsf{T}_2, \mathsf{T}_3)$ est dans $\nu\mathsf{MeetJoin}$, et $\mathsf{T}_1 \sqcup \mathsf{T}_2 = \mathsf{T}_3$ pour le triplet $\sqcup(\mathsf{T}_1, \mathsf{T}_2, \mathsf{T}_3)$. Cette notation laisse deviner que ces opérateurs sont des fonctions, c'est-à-dire que le T_3 associé à un T_1 et un T_2 est unique, résultat qui découlera naturellement des propriétés que nous allons montrer. Par simplicité

d'écriture, on utilise dès maintenant cette notation. Prouvons maintenant que $\sqcap$ et $\sqcup$ coïncident avec les bornes inférieure et supérieure, c'est-à-dire le plus grand sous-type commun et le plus petit sur-type commun. Ces propriétés sont ici énoncées uniquement pour la borne inférieure, qui présente plus d'intérêt pour le Dπ-calcul, mais les preuves portent bien évidemment en même temps sur la borne supérieure, sans quoi elles ne pourraient pas être vraies en présence de constructions contravariantes. La première proposition indique que $\sqcap$ et $\sqcup$ sont symétriques en leurs deux premiers arguments, ce qui facilitera les preuves par la suite.

Proposition 2.10 ($\sqcap$ est symétrique)**.** *Quels que soient les types* T_1*,* T_2 *et* T_3*,* $\mathsf{T}_1 \sqcap \mathsf{T}_2 = \mathsf{T}_3$ *si et seulement si* $\mathsf{T}_2 \sqcap \mathsf{T}_1 = \mathsf{T}_3$*.*

Démonstration. La preuve repose simplement sur le fait que la relation

$$\begin{aligned}\mathcal{R} = \{&_\sqcap(\mathsf{T}_1, \mathsf{T}_2, \mathsf{T}_3) \mid {}_\sqcap(\mathsf{T}_2, \mathsf{T}_1, \mathsf{T}_3) \in \nu\mathsf{MeetJoin}\} \\ &\cup \{{}_\sqcup(\mathsf{T}_1, \mathsf{T}_2, \mathsf{T}_3) \mid {}_\sqcup(\mathsf{T}_2, \mathsf{T}_1, \mathsf{T}_3) \in \nu\mathsf{MeetJoin}\}\end{aligned}$$

est un post-point fixe de la fonction $\mathsf{MeetJoin}$. Ce résultat est assez immédiat dans la mesure où la définition de $\mathsf{MeetJoin}$ est rigoureusement symétrique. □

La preuve que $\sqcap$ et $\sqcup$ sont les bornes recherchées peut se décomposer en deux propositions : tout d'abord le fait que $\mathsf{T}_1 \sqcap \mathsf{T}_2 = \mathsf{T}_3$ implique que T_3 soit un sous-type de T_1 et de T_2 ; puis le fait qu'il est plus grand que tout sous-type commun.

Proposition 2.11 ($\sqcap$ calcule un sous-type)**.** *Pour tous types* T_1*,* T_2 *et* T_3 *tels que* $\mathsf{T}_1 \sqcap \mathsf{T}_2 = \mathsf{T}_3$*, on a* $\mathsf{T}_3 <: \mathsf{T}_1$ *et* $\mathsf{T}_3 <: \mathsf{T}_2$*.*

Démonstration. Considérons la relation

$$\mathcal{R} = \nu\mathsf{Sub} \cup \{(\mathsf{T}_1, \mathsf{T}_2) \mid \exists \mathsf{T}_3, \mathsf{T}_2 \sqcap \mathsf{T}_3 = \mathsf{T}_1 \text{ ou } \mathsf{T}_1 \sqcup \mathsf{T}_3 = \mathsf{T}_2\}$$

et prouvons qu'il s'agit d'un post-point fixe de Sub. La symétrie des opérateurs $\sqcap$ et $\sqcup$ finira la preuve.

Soit un couple $(\mathsf{T}_1, \mathsf{T}_2)$ dans cet ensemble $\mathcal{R}$. S'il est dans $\nu\mathsf{Sub}$, le résultat est immédiat. Sinon, soit T_3 tel que $\mathsf{T}_2 \sqcap \mathsf{T}_3 = \mathsf{T}_1$ ou $\mathsf{T}_1 \sqcup \mathsf{T}_3 = \mathsf{T}_2$. En raisonnant sur le cas de la définition de $\mathsf{MeetJoin}$ correspondant à cette propriété, on peut prouver que $(\mathsf{T}_1, \mathsf{T}_2)$ est dans $\mathsf{Sub}(\mathcal{R})$. Par exemple, si le cas est $\mathsf{r}\langle \mathsf{T}_2^r \rangle \sqcap \mathsf{w}\langle \mathsf{T}_3^w \rangle = \mathsf{rw}\langle \mathsf{T}_2^r, \mathsf{T}_3^w \rangle = \mathsf{T}_1$, on peut déduire que $\mathsf{T}_3^w <: \mathsf{T}_2^r$. La réflexivité de $<:$ garantit par ailleurs que $\mathsf{T}_2^r <: \mathsf{T}_2^r$. D'où l'on déduit que $(\mathsf{T}_1, \mathsf{T}_2) \in \mathsf{Sub}(\nu\mathsf{Sub}) \subseteq \mathsf{Sub}(\mathcal{R})$. Les autres cas se prouvent de façon similaire. □

Proposition 2.12 ($\sqcap$ est le plus grand sous-type)**.** *Pour tous types* T_1*,* T_2*,* T_3 *et* T *tels que* $\mathsf{T}_1 \sqcap \mathsf{T}_2 = \mathsf{T}_3$*,* $\mathsf{T} <: \mathsf{T}_1$ *et* $\mathsf{T} <: \mathsf{T}_2$*, on a* $\mathsf{T} <: \mathsf{T}_3$*.*

Démonstration. Prouvons que la relation

$$\begin{aligned}\mathcal{R} = \quad &\{(\mathsf{T}, \mathsf{T}_3) \mid \exists \mathsf{T}_1, \mathsf{T}_2 \text{ tels que } \mathsf{T} <: \mathsf{T}_1, \mathsf{T} <: \mathsf{T}_2, \mathsf{T}_1 \sqcap \mathsf{T}_2 = \mathsf{T}_3\} \\ \cup \quad &\{(\mathsf{T}_3, \mathsf{T}) \mid \exists \mathsf{T}_1, \mathsf{T}_2 \text{ tels que } \mathsf{T}_1 <: \mathsf{T}, \mathsf{T}_2 <: \mathsf{T}, \mathsf{T}_1 \sqcup \mathsf{T}_2 = \mathsf{T}_3\} \\ \cup \quad &\nu\mathsf{Sub}\end{aligned}$$

est un post-point fixe de Sub en considérant un couple de $\mathcal{R}$. Ce couple est donc dans l'une des trois composantes formant $\mathcal{R}$. Dans les deux premières possibilités, nous étudions tous les cas possibles pour la condition $\mathsf{T}_1 \sqcap \mathsf{T}_2 = \mathsf{T}_3$ ou $\mathsf{T}_1 \sqcup \mathsf{T}_2 = \mathsf{T}_3$, ce qui revient à envisager tous les cas de la définition de $\mathsf{MeetJoin}$. Regardons quelques cas typiques.

- $\mathsf{rw}\langle \mathsf{T}_1^r, \mathsf{T}_3^w\rangle \sqcap \mathsf{r}\langle \mathsf{T}_2^r\rangle = \mathsf{rw}\langle \mathsf{T}_3^r, \mathsf{T}_3^w\rangle$ alors nous savons que $\mathsf{T}_1^r \sqcap \mathsf{T}_2^r = \mathsf{T}_3^r$. $\mathsf{T} <: \mathsf{T}_1$ implique que T soit de la forme $\mathsf{rw}\langle \mathsf{T}^r, \mathsf{T}^w\rangle$, avec $\mathsf{T}_3^w <: \mathsf{T}^w <: \mathsf{T}^r <: \mathsf{T}_1^r$. $\mathsf{T} <: \mathsf{T}_2$ donne alors $\mathsf{T}^r <: \mathsf{T}_2^r$, ce qui permet de conclure que $\mathsf{T}^r \mathcal{R} \mathsf{T}_3^r$. Ainsi $\mathsf{T}_3^w \mathcal{R} \mathsf{T}^w \mathcal{R} \mathsf{T}^r \mathcal{R} \mathsf{T}_3^r$ donne le résultat $\mathsf{T}\ \mathsf{Sub}(\mathcal{R})\ \mathsf{T}_3$.
- $\mathsf{rw}\langle \mathsf{T}_1^r, \mathsf{T}_1^w\rangle \sqcup \mathsf{rw}\langle \mathsf{T}_2^r, \mathsf{T}_2^w\rangle = \mathsf{r}\langle \mathsf{T}_3^r\rangle$ alors nous savons $\mathsf{T}_1^w \not\downarrow \mathsf{T}_2^w$ et $\mathsf{T}_1^r \sqcup \mathsf{T}_2^r = \mathsf{T}_3^r$. $\mathsf{T}_1 <: \mathsf{T}$ implique que T soit lui aussi un type de canal. Supposons que T soit $\mathsf{rw}\langle \mathsf{T}^r, \mathsf{T}^w\rangle$ ou $\mathsf{w}\langle \mathsf{T}^w\rangle$. Dans ces deux cas, $\mathsf{T}_1 <: \mathsf{T}$ et $\mathsf{T}_2 <: \mathsf{T}$ impliquerait $\mathsf{T}^w <: \mathsf{T}_1^w$ et $\mathsf{T}^w <: \mathsf{T}_2^w$, ce qui contredit l'hypothèse d'incompatibilité des types T_1^w et T_2^w. Nous pouvons en conclure que T doit être $\mathsf{r}\langle \mathsf{T}^r\rangle$, avec $\mathsf{T}_1^r <: \mathsf{T}^r$ et $\mathsf{T}_2^r <: \mathsf{T}^r$ soit $\mathsf{T}_3^r \mathcal{R} \mathsf{T}^r$ ce qui nous donne $\mathsf{T}_3\ \mathsf{Sub}(\mathcal{R})\ \mathsf{T}$.
- $\mathsf{rw}\langle \mathsf{T}_1^r, \mathsf{T}_1^w\rangle \sqcup \mathsf{rw}\langle \mathsf{T}_2^r, \mathsf{T}_2^w\rangle = \mathsf{rw}\langle \mathsf{T}_3^r, \mathsf{T}_3^w\rangle$ avec $\mathsf{T}_1^r \sqcup \mathsf{T}_2^r = \mathsf{T}_3^r$, $\mathsf{T}_1^w \sqcap \mathsf{T}_2^w = \mathsf{T}_3^w$ et $\mathsf{T}_1^w <: \mathsf{T}_1^r$. D'après la proposition 2.11, nous pouvons en déduire $\mathsf{T}_3^w <: \mathsf{T}_1^w$ et $\mathsf{T}_1^r <: \mathsf{T}_3^r$. Par transitivité, nous obtenons $\mathsf{T}_3^w <: \mathsf{T}_3^r$.
  T peut prendre les formes $\mathsf{r}\langle \mathsf{T}^r\rangle$, $\mathsf{w}\langle \mathsf{T}^w\rangle$ ou $\mathsf{rw}\langle \mathsf{T}^r, \mathsf{T}^w\rangle$. Par exemple dans le troisième cas, $\mathsf{T}_1 <: \mathsf{T}$ et $\mathsf{T}_2 <: \mathsf{T}$ permettent de conclure $\mathsf{T}_1^r <: \mathsf{T}^r$, $\mathsf{T}_2^r <: \mathsf{T}^r$, donc $\mathsf{T}_3^r \mathcal{R} \mathsf{T}^r$, et $\mathsf{T}^w <: \mathsf{T}_1^w$ et $\mathsf{T}^w <: \mathsf{T}_2^w$ donc $\mathsf{T}^w \mathcal{R} \mathsf{T}_3^w$. Nous avons donc réuni toutes les hypothèses pour conclure que $\mathsf{T}_3\ \mathsf{Sub}(\mathcal{R})\ \mathsf{T}$.

Si le couple est dans $\nu\mathsf{Sub}$, le résultat est immédiat. Nous avons donc prouvé que $\mathcal{R} \subseteq \nu\mathsf{Sub}$. □

Les résultats précédents permettent de voir que $\sqcap$ et $\sqcup$ sont effectivement les opérateurs de bornes inférieure et supérieure engendrés par l'ordre $<:$. En particulier il s'agit bien de fonctions puisque $\mathsf{T}_1 \sqcap \mathsf{T}_2 = \mathsf{T}_3$ et $\mathsf{T}_1 \sqcap \mathsf{T}_2 = \mathsf{T}_4$ impliquent $\mathsf{T}_3 <: \mathsf{T}_4$ et $\mathsf{T}_4 <: \mathsf{T}_3$, c'est-à-dire $\mathsf{T}_3 = \mathsf{T}_4$.

La notion de borne inférieure de deux types étant maintenant bien établie, revenons à la question : à quelle condition cette borne inférieure existe-t-elle ? Nous allons voir que la compatibilité des types est suffisante pour assurer son existence. Cette condition permettra notamment de garantir la cohérence des connaissances collectées par un processus à propos d'un nom donné.

La preuve que la compatibilité de deux types suffit se décompose en deux parties : on définira tout d'abord un type arborescent en donnant formellement les étiquettes de chacun de ses nœuds à partir de celles des deux types dont on cherche la borne inférieure ; puis on montrera que ce type coinductif est effectivement leur borne inférieure. Pour décrire correctement le type, définissons tout d'abord les positions d'un arbre (de type) en donnant le sous-arbre enraciné à cette position.

Définition 2.13 (Sous-arbre situé à une position)**.** *Quand p est un mot dont les lettres peuvent être r, w ou un nombre, le sous-arbre situé à la position p d'un type arborescent T, ce que l'on note $\mathsf{T}|_p$, est défini par cas sur p.*

- $\mathsf{T}|_\varepsilon = \mathsf{T}$ *où ε est la position vide.*
- $\mathsf{loc}|_p$ *n'est pas défini si $p \neq \varepsilon$.*
- $\mathsf{r}\langle \mathsf{T}^r\rangle|_{rp} = \mathsf{T}^r|_p$.
- $\mathsf{w}\langle \mathsf{T}^w\rangle|_{wp} = \mathsf{T}^w|_p$.
- $\mathsf{rw}\langle \mathsf{T}^r, \mathsf{T}^w\rangle|_{rp} = \mathsf{T}^r|_p$.
- $\mathsf{rw}\langle \mathsf{T}^r, \mathsf{T}^w\rangle|_{wp} = \mathsf{T}^w|_p$.
- $\mathsf{C}_{@s}|_{0p} = \mathsf{C}|_p$.
- $(\mathsf{T}_1, \ldots, \mathsf{T}_n)|_{ip} = \mathsf{T}_i|_p$ *pour* $1 \leq i \leq n$.
- $\sum \vec{x} : \vec{\mathsf{loc}}.\ \mathsf{T}\Big|_{0p} = \mathsf{T}|_p$.

Définition 2.14 (Ensemble des positions d'un type)**.** *L'ensemble des positions d'un type* T *est l'ensemble des positions* p *telles que* $\mathsf{T}|_p$ *soit défini.*

On voit aisément que l'ensemble des positions d'un type donné est forcément clos par préfixe, c'est-à-dire que, pour toute position p d'un type T, tous les préfixes de p sont aussi des positions de T. On voit aussi que l'ensemble des positions de types équivalents sont égaux, car les positions sont des mots de longueur finie, ainsi que le fait que les sous-arbres situés à une même position dans deux types équivalents sont toujours équivalents.

La contravariance du constructeur de type $\mathsf{w}\langle\cdot\rangle$ et la covariance des autres induisent une variance associée à une occurrence donnée.

Définition 2.15 (Variance d'une position)**.** *Une position* p *est dite* covariante *si elle contient un nombre pair de* w *et* contravariante *dans le cas contraire.*

Proposition 2.16 (Sous-typage rapporté à une position)**.** *Soient deux types* T_1 *et* T_2 *tels que* $\mathsf{T}_1 <: \mathsf{T}_2$ *et* p *une position commune des ensembles de positions de* T_1 *et* T_2*. Alors :*

- $\mathsf{T}_1|_p <: \mathsf{T}_2|_p$ *si* p *est covariante ;*
- $\mathsf{T}_2|_p <: \mathsf{T}_1|_p$ *si* p *est contravariante.*

Démonstration. Cette proposition se prouve par une simple induction sur la longueur de la position p. En effet, si p est ε, le résultat est immédiat. Sinon, considérons par exemple le cas où p est de la forme wp'. Alors les deux types T_1 et T_2 doivent être de la forme $\mathsf{w}\langle\mathsf{T}^w\rangle$ ou $\mathsf{rw}\langle\mathsf{T}^r, \mathsf{T}^w\rangle$. Dans les deux cas on peut en déduire $\mathsf{T}_2^w <: \mathsf{T}_1^w$. La variance de p' étant l'opposée de celle de p, on obtient simplement le résultat par hypothèse d'induction. Les autres cas sont similaires. □

On utilisera ces différentes notions pour prouver le théorème majeur de la théorie des types coinductifs, à savoir le théorème de complétude sous condition de l'ensemble des types.

Théorème 2.17 (Complétude sous condition)**.** *Deux types quelconques sont* $\downarrow$*-compatibles si et seulement si leur borne inférieure est définie.*

Démonstration. La direction « si » est un corollaire immédiat de la proposition 2.11. Regardons donc la direction « seulement si ». Cette preuve se décompose en deux étapes majeures : la construction d'un candidat de borne inférieure pour deux types donnés ; la preuve que ce candidat est la borne recherchée.

Considérons deux types T_1 et T_2. La construction d'un candidat pour leur borne inférieure repose sur les notions de positions dans les types. On pourrait chercher à la décrire via son ensemble de positions, que l'on déduirait de ceux de T_1 et T_2. Cependant, le calcul de toutes les positions apparaissant effectivement dans la borne inférieure serait assez fastidieux. Par exemple dans le cas $\mathsf{w}\langle\mathsf{T}_1^w\rangle \sqcap \mathsf{r}\langle\mathsf{T}_2^r\rangle$ il s'agira simplement de l'union des positions de T_1 et T_2. Mais dans le cas un peu plus complexe de la borne inférieure

$$\mathsf{w}\langle\mathsf{rw}\langle\mathsf{r}\langle\mathsf{rw}\langle\rangle\rangle, \mathsf{rw}\langle\mathsf{rw}\langle\rangle\rangle\rangle\rangle \sqcap \mathsf{w}\langle\mathsf{rw}\langle\mathsf{r}\langle\mathsf{r}\langle\rangle\rangle, \mathsf{rw}\langle\mathsf{r}\langle\rangle\rangle\rangle\rangle$$

qui se trouve être simplement $\mathsf{w}\langle\mathsf{r}\langle\mathsf{r}\langle\mathsf{r}\langle\rangle\rangle\rangle\rangle$ car les composantes de ces types $\mathsf{rw}\langle\mathsf{rw}\langle\rangle\rangle$ et $\mathsf{rw}\langle\mathsf{r}\langle\rangle\rangle$ sont incompatibles, de nombreuses occurrences communes sont élaguées. Ces exemples montrent alors une autre approche possible : dans

le premier, il suffit de regarder les deux racines des types pour savoir quelle est la seule borne inférieure possible ; dans le second, il faut étudier non seulement la racine mais aussi poursuivre l'investigation aux positions w, ww, wr et les autres positions de préfixe wr. Dans les deux cas, une investigation au niveau des positions communes aux deux types sera suffisante pour composer la borne inférieure ou supérieure : dans le premier cas, il suffit de vérifier à la racine que le type en émission est un sous-type du type en réception ; dans le second, c'est au niveau de la position commune ww que l'incompatibilité entre $\mathsf{rw}\langle\mathsf{rw}\langle\rangle\rangle$ et $\mathsf{rw}\langle\mathsf{r}\langle\rangle\rangle$ apparaît.

On parlera donc de l'ensemble des positions $\mathcal{P}_{\mathsf{T}_1\sqcap\mathsf{T}_2}$ auxquelles la borne inférieure de T_1 et T_2 doit être « étudiée » comme l'ensemble des p telles que p soit une position commune de T_1 et T_2 et pour tout préfixe p' de p :

– si p' est covariante, $\mathsf{T}_1|_{p'} \downarrow \mathsf{T}_2|_{p'}$;
– si p' est contravariante, $\mathsf{T}_1|_{p'} \uparrow \mathsf{T}_2|_{p'}$.

Cet ensemble est forcément clos par préfixe.

On notera $\mathsf{T}_{\mathsf{T}_1\sqcap\mathsf{T}_2}$ le pré-type arborescent candidat à être la borne inférieure. On se contentera, pour le définir, de considérer les types T_1 et T_2 aux positions de $\mathcal{P}_{\mathsf{T}_1\sqcap\mathsf{T}_2}$. Le « choix » du constructeur de type situé à la position $p \in \mathcal{P}_{\mathsf{T}_1\sqcap\mathsf{T}_2}$ dans $\mathsf{T}_{\mathsf{T}_1\sqcap\mathsf{T}_2}$ est directement calqué sur la définition de la fonction $\mathsf{MeetJoin}$. In extenso, pour une position p de $\mathcal{P}_{\mathsf{T}_1\sqcap\mathsf{T}_2}$, on raisonne par cas sur les formes de $\mathsf{T}_1|_p$ et $\mathsf{T}_2|_p$ (bien entendu, quand $\mathsf{T}_1|_p$ et $\mathsf{T}_2|_p$ n'ont pas la même forme, on ne considère le cas qu'une seule fois, la définition devant être symétrique) :

$\mathsf{r}\langle\mathsf{T}_1^r\rangle$, $\mathsf{r}\langle\mathsf{T}_2^r\rangle$ le nœud de $\mathsf{T}_{\mathsf{T}_1\sqcap\mathsf{T}_2}$ à la position p est $\mathsf{r}\langle\cdot\rangle$; que p soit covariante ou contravariante, la condition de compatibilité des types $\mathsf{r}\langle\mathsf{T}_1^r\rangle$ et $\mathsf{r}\langle\mathsf{T}_2^r\rangle$ implique que T_1^r et T_2^r vérifient la même condition de compatibilité et pr est évidemment une position commune des types T_1 et T_2, donc elle doit être dans $\mathcal{P}_{\mathsf{T}_1\sqcap\mathsf{T}_2}$;

$\mathsf{w}\langle\mathsf{T}_1^w\rangle$, $\mathsf{w}\langle\mathsf{T}_2^w\rangle$ le nœud de $\mathsf{T}_{\mathsf{T}_1\sqcap\mathsf{T}_2}$ à la position p est $\mathsf{w}\langle\cdot\rangle$; on peut faire une remarque similaire à celle du cas précédent quant à la position pw, à un changement de variance près ;

$\mathsf{r}\langle\mathsf{T}_1^r\rangle$, $\mathsf{w}\langle\mathsf{T}_2^w\rangle$ p est covariante car ces deux types ne sont pas $\uparrow$-compatibles ; $\mathsf{T}_{\mathsf{T}_1\sqcap\mathsf{T}_2}|_p$ doit être $\mathsf{rw}\langle\mathsf{T}_1^r, \mathsf{T}_2^w\rangle$; la $\downarrow$-compatibilité de $\mathsf{r}\langle\mathsf{T}_1^r\rangle$ et $\mathsf{w}\langle\mathsf{T}_2^w\rangle$ garantit l'existence d'un type de la forme $\mathsf{rw}\langle\mathsf{T}^r, \mathsf{T}^w\rangle$ tel que $\mathsf{T}_2^w <: \mathsf{T}^w <: \mathsf{T}^r <: \mathsf{T}_1^r$ donc $\mathsf{rw}\langle\mathsf{T}_1^r, \mathsf{T}_2^w\rangle$ est un type bien formé ;

$\mathsf{rw}\langle\mathsf{T}_1^r, \mathsf{T}_1^w\rangle$, $\mathsf{r}\langle\mathsf{T}_2^r\rangle$, p covariante $\mathsf{T}_{\mathsf{T}_1\sqcap\mathsf{T}_2}|_p$ doit être de la forme $\mathsf{rw}\langle\cdot, \mathsf{T}_1^w\rangle$; la condition de compatibilité assure que T_1^r et T_2^r sont aussi compatibles, or pr est obligatoirement une position commune de T_1 et T_2 ce qui permet de conclure que pr est dans $\mathcal{P}_{\mathsf{T}_1\sqcap\mathsf{T}_2}$;

$\mathsf{rw}\langle\mathsf{T}_1^r, \mathsf{T}_1^w\rangle$, $\mathsf{r}\langle\mathsf{T}_2^r\rangle$, p contravariante $\mathsf{T}_{\mathsf{T}_1\sqcap\mathsf{T}_2}|_p$ doit être de la forme $\mathsf{r}\langle\cdot\rangle$; pour la même raison que précédemment, pr doit être dans $\mathcal{P}_{\mathsf{T}_1\sqcap\mathsf{T}_2}$;

$\mathsf{rw}\langle\mathsf{T}_1^r, \mathsf{T}_1^w\rangle$, $\mathsf{w}\langle\mathsf{T}_2^w\rangle$ quelle que soit la variance de p, le raisonnement est calqué sur les deux cas précédents ;

$\mathsf{rw}\langle\mathsf{T}_1^r, \mathsf{T}_1^w\rangle$, $\mathsf{rw}\langle\mathsf{T}_2^r, \mathsf{T}_2^w\rangle$, p est covariante $\mathsf{T}_{\mathsf{T}_1\sqcap\mathsf{T}_2}|_p$ doit être de la forme $\mathsf{rw}\langle\cdot,\cdot\rangle$ et les deux positions pr et pw doivent être dans $\mathcal{P}_{\mathsf{T}_1\sqcap\mathsf{T}_2}$;

$\mathsf{rw}\langle\mathsf{T}_1^r, \mathsf{T}_1^w\rangle$, $\mathsf{rw}\langle\mathsf{T}_2^r, \mathsf{T}_2^w\rangle$, p est contravariante trois cas se dégagent :

1. $\mathsf{T}_1^r \uparrow \mathsf{T}_2^r$ et $\mathsf{T}_1^w \downarrow \mathsf{T}_2^w$: $\mathsf{T}_{\mathsf{T}_1\sqcap\mathsf{T}_2}|_p$ doit être de la forme $\mathsf{rw}\langle\cdot,\cdot\rangle$, et les deux positions pr et pw doivent être dans $\mathcal{P}_{\mathsf{T}_1\sqcap\mathsf{T}_2}$;

2. $\mathsf{T}_1^r \not\downarrow \mathsf{T}_2^r$ et $\mathsf{T}_1^w \downarrow \mathsf{T}_2^w$, $\mathsf{T}_{\mathsf{T}_1 \sqcap \mathsf{T}_2}|_p$ doit être de la forme $\mathsf{w}\langle\cdot\rangle$ et seule la position pw doit être dans $\mathcal{P}_{\mathsf{T}_1 \sqcap \mathsf{T}_2}$;
3. $\mathsf{T}_1^r \uparrow \mathsf{T}_2^r$ et $\mathsf{T}_1^w \not\downarrow \mathsf{T}_2^w$, $\mathsf{T}_{\mathsf{T}_1 \sqcap \mathsf{T}_2}|_p$ doit être de la forme $\mathsf{r}\langle\cdot\rangle$ et seule la position pr doit être dans $\mathcal{P}_{\mathsf{T}_1 \sqcap \mathsf{T}_2}$;

loc, loc ; $(\vec{\mathsf{T}_1})$**,** $(\vec{\mathsf{T}_2})$ **;** $\sum \vec{x} : \vec{\mathbf{loc}}.\ \mathsf{T}'_1$**,** $\sum \vec{x} : \vec{\mathbf{loc}}.\ \mathsf{T}'_2$ **;** $\mathsf{C}'_{1@s}$ **et** $\mathsf{C}'_{2@s}$ Ces quatre cas sont tous semblables : $\mathsf{T}_{\mathsf{T}_1 \sqcap \mathsf{T}_2}|_p$ doit avoir le même constructeur de tête que ces types et les positions filles doivent toutes être dans $\mathcal{P}_{\mathsf{T}_1 \sqcap \mathsf{T}_2}$;

Si $\mathsf{T}_1|_p$ et $\mathsf{T}_2|_p$ ne correspondent à aucun des cas cités jusqu'à présent, ils ne peuvent pas être compatibles.

L'ensemble des positions auxquelles $\mathsf{T}_{\mathsf{T}_1 \sqcap \mathsf{T}_2}$ vient d'être défini est clos par préfixe (en particulier il contient ε), comme nous l'avons déjà remarqué, ce qui permet de garantir que l'on obtient bien un arbre. Par ailleurs, tous les nœuds de cet arbre ont une étiquette parmi toutes celles autorisées dans la syntaxe des pré-types et ont exactement l'arité correspondante à cette étiquette. En effet, à chaque fois qu'une étiquette a été attribuée dans cette définition, on a soit défini immédiatement les différents fils nécessaires pour le nœud (par exemple dans le cas $(\mathsf{r}\langle\mathsf{T}_1^r\rangle, \mathsf{w}\langle\mathsf{T}_2^w\rangle)$ où les seuls fils possibles sont déjà connus) soit vérifié que les positions de ces fils étaient dans l'ensemble $\mathcal{P}_{\mathsf{T}_1 \sqcap \mathsf{T}_2}$ des positions de $\mathsf{T}_{\mathsf{T}_1 \sqcap \mathsf{T}_2}$ qui sont définies. On a donc bien défini un pré-type.

Il faut maintenant montrer qu'il s'agit bien là d'un type. Remarquons tout d'abord que toutes les variables qu'il contient doivent aussi apparaître dans les types T_1 et T_2, où elles doivent être liées. Or ces lieurs dans T_1 et T_2 sont appariés à un lieur pour la même variable dans $\mathsf{T}_{\mathsf{T}_1 \sqcap \mathsf{T}_2}$, ce qui garantit que toutes les variables sont liées dans ce pré-type.

Prouvons donc que tous les $\mathsf{rw}\langle\mathsf{T}^r, \mathsf{T}^w\rangle$ que ce pré-type contient sont tels que $\mathsf{T}^w <: \mathsf{T}^r$. Pour cela, montrons que la relation $\mathcal{R}^+$ est une relation de sous-typage, où $\mathcal{R}$ est définie ainsi :

$$\begin{aligned}
\mathcal{R} = \quad & \{(\mathsf{T}, \mathsf{T}_{\mathsf{T}_1 \sqcap \mathsf{T}_2}|_p), (\mathsf{T}_{\mathsf{T}_1 \sqcap \mathsf{T}_2}|_p\,, \mathsf{T}_1|_p), (\mathsf{T}_{\mathsf{T}_1 \sqcap \mathsf{T}_2}|_p\,, \mathsf{T}_2|_p) \\
& \quad \mid p \in \mathcal{P}_{\mathsf{T}_1 \sqcap \mathsf{T}_2} \text{ covariante, } \mathsf{T} \text{ type minorant de } \{\mathsf{T}_1|_p\,, \mathsf{T}_2|_p\}\} \\
\cup \quad & \{(\mathsf{T}_{\mathsf{T}_1 \sqcap \mathsf{T}_2}|_p\,, \mathsf{T}), (\mathsf{T}_1|_p\,, \mathsf{T}_{\mathsf{T}_1 \sqcap \mathsf{T}_2}|_p), (\mathsf{T}_2|_p\,, \mathsf{T}_{\mathsf{T}_1 \sqcap \mathsf{T}_2}|_p) \\
& \quad \mid p \in \mathcal{P}_{\mathsf{T}_1 \sqcap \mathsf{T}_2} \text{ contravariante, } \mathsf{T} \text{ type majorant de } \{\mathsf{T}_1|_p\,, \mathsf{T}_2|_p\}\} \\
\cup \quad & \nu\mathsf{Sub}
\end{aligned}$$

Pour cela, on peut prouver que $\mathcal{R}$ est incluse dans $\mathsf{Sub}(\mathcal{R}^+)$. La proposition 2.6 pourra alors être utilisée pour en déduire $\mathcal{R}^+ \subseteq \mathsf{Sub}(\mathcal{R}^+)$, ce qui implique $\mathcal{R}^+ \subseteq \nu\mathsf{Sub}$.

Considérons donc un couple dans la relation $\mathcal{R}$. Si ce couple est dans la partie $\nu\mathsf{Sub}$ de $\mathcal{R}$, le résultat est immédiat. Dans le cas contraire, on sait que le couple doit correspondre à l'un des deux autres ensembles qui composent la relation $\mathcal{R}$, en particulier correspondre à une position p. On peut alors considérer d'un seul coup tous les couples rajoutés pour cette position particulière et on raisonne suivant la forme de $\mathsf{T}_1|_p$ et $\mathsf{T}_2|_p$, comme lors de la définition de $\mathsf{T}_{\mathsf{T}_1 \sqcap \mathsf{T}_2}$, puisque cette vérification lui correspond point par point. Parmi tous les cas, contentons-nous de prendre un exemple typique : $\mathsf{T}_1|_p = \mathsf{rw}\langle\mathsf{T}_1^r, \mathsf{T}_1^w\rangle$ et $\mathsf{T}_2|_p = \mathsf{r}\langle\mathsf{T}_2^r\rangle$. Si p est covariante, $\mathsf{T}_{\mathsf{T}_1 \sqcap \mathsf{T}_2}|_p$ se voit attribuer la forme $\mathsf{rw}\langle\cdot, \mathsf{T}_1^w\rangle$ avec pr une position de l'ensemble $\mathcal{P}_{\mathsf{T}_1 \sqcap \mathsf{T}_2}$. Par ailleurs, puisque T est un minorant de $\mathsf{T}_1|_p$ et $\mathsf{T}_2|_p$, il doit être de la forme $\mathsf{rw}\langle\mathsf{T}^r, \mathsf{T}^w\rangle$ avec $\mathsf{T}^r <: \mathsf{T}_1^r$, $\mathsf{T}^r <: \mathsf{T}_2^r$ et $\mathsf{T}_1^w <: \mathsf{T}^w$. Les deux premières inégalités permettent de conclure, avec le fait que la position

pr appartient à $\mathcal{P}_{\mathsf{T}_1\sqcap\mathsf{T}_2}$, que $(\mathsf{T}^r, \mathsf{T}_{\mathsf{T}_1\sqcap\mathsf{T}_2}|_{pr})$ appartient à $\mathcal{R}$. La troisième, avec le fait que T est un type donc $\mathsf{T}^w <: \mathsf{T}^r$, permet d'en déduire $(\mathsf{T}, \mathsf{T}_{\mathsf{T}_1\sqcap\mathsf{T}_2}|_p) \in \mathsf{Sub}(\mathcal{R}) \subseteq \mathsf{Sub}(\mathcal{R}^+)$. Par ailleurs, pour prouver que $(\mathsf{T}_{\mathsf{T}_1\sqcap\mathsf{T}_2}|_p\,, \mathsf{T}_1|_p)$ est dans $\mathsf{Sub}(\mathcal{R}^+)$, on doit avoir dans $\mathcal{R}^+$ les couples $(\mathsf{T}_1|_{pw}\,, \mathsf{T}_{\mathsf{T}_1\sqcap\mathsf{T}_2}|_{pw}) = (\mathsf{T}_1^w, \mathsf{T}_1^w)$, $(\mathsf{T}_{\mathsf{T}_1\sqcap\mathsf{T}_2}|_{pw}\,, \mathsf{T}_{\mathsf{T}_1\sqcap\mathsf{T}_2}|_{pr})$ et $(\mathsf{T}_{\mathsf{T}_1\sqcap\mathsf{T}_2}|_{pr}\,, \mathsf{T}_1^r)$. Le premier y est par $\nu\mathsf{Sub}$. Le deuxième vient du minorant $\mathsf{T} = \mathsf{rw}\langle\mathsf{T}^r, \mathsf{T}^w\rangle$ de $\mathsf{T}_1|_p$ et $\mathsf{T}_2|_p$. En effet, comme nous l'avons vu, on obtient dans ce cas-là : $\mathsf{T}_1^w <: \mathsf{T}^w$ et $\mathsf{T}^r\mathcal{R}\,\mathsf{T}_{\mathsf{T}_1\sqcap\mathsf{T}_2}|_{pr}$. De plus T est un type donc $\mathsf{T}^w <: \mathsf{T}^r$. On peut donc conclure que $\mathsf{T}_1^w\ \mathcal{R}^+\ \mathsf{T}_{\mathsf{T}_1\sqcap\mathsf{T}_2}|_{pr}$. Enfin, le troisième, $(\mathsf{T}_{\mathsf{T}_1\sqcap\mathsf{T}_2}|_{pr}\,, \mathsf{T}_1^r)$, vient du fait que pr soit dans $\mathcal{P}_{\mathsf{T}_1\sqcap\mathsf{T}_2}$. Ce qui nous permet de conclure que $(\mathsf{T}_{\mathsf{T}_1\sqcap\mathsf{T}_2}|_p\,, \mathsf{T}_1|_p)$ est dans $\mathsf{Sub}(\mathcal{R}^+)$. Enfin, pour prouver que $(\mathsf{T}_{\mathsf{T}_1\sqcap\mathsf{T}_2}|_p\,, \mathsf{T}_2|_p)$ est dans $\mathsf{Sub}(\mathcal{R}^+)$, il suffit de prouver que $(\mathsf{T}_{\mathsf{T}_1\sqcap\mathsf{T}_2}|_{pr}\,, \mathsf{T}_2^r)$ est dans $\mathcal{R}$, ce qui découle directement de $pr \in \mathcal{P}_{\mathsf{T}_1\sqcap\mathsf{T}_2}$.

Si p est contravariante, on sait que pr est aussi dans $\mathcal{P}_{\mathsf{T}_1\sqcap\mathsf{T}_2}$, ce qui permet de conclure que $(\mathsf{T}_1|_{pr}\,, \mathsf{T}_{\mathsf{T}_1\sqcap\mathsf{T}_2}|_{pr})$ et $(\mathsf{T}_2|_{pr}\,, \mathsf{T}_{\mathsf{T}_1\sqcap\mathsf{T}_2}|_{pr})$ sont dans $\mathcal{R}$. On en déduit facilement que $(\mathsf{T}_1|_p\,, \mathsf{T}_{\mathsf{T}_1\sqcap\mathsf{T}_2}|_p)$ et $(\mathsf{T}_2|_p\,, \mathsf{T}_{\mathsf{T}_1\sqcap\mathsf{T}_2}|_p)$ sont dans $\mathsf{Sub}(\mathcal{R}) \subseteq \mathsf{Sub}(\mathcal{R}^+)$. De même, si T est un majorant de $\mathsf{T}_1|_p$ et $\mathsf{T}_2|_p$, il doit être de la forme $\mathsf{r}\langle\mathsf{T}^r\rangle$ où T^r est un majorant de $\mathsf{T}_1|_{pr}$ et $\mathsf{T}_2|_{pr}$. D'où l'on déduit que $(\mathsf{T}_{\mathsf{T}_1\sqcap\mathsf{T}_2}|_{pr}\,, \mathsf{T}^r)$ est dans $\mathcal{R}$, donc $(\mathsf{T}_{\mathsf{T}_1\sqcap\mathsf{T}_2}|_p\,, \mathsf{T})$ dans $\mathsf{Sub}(\mathcal{R})$, ce qui achève de prouver que $\mathcal{R}$ est incluse dans $\mathsf{Sub}(\mathcal{R}^+)$.

Considérons maintenant une position p de $\mathsf{T}_{\mathsf{T}_1\sqcap\mathsf{T}_2}$ de la forme $\mathsf{rw}\langle\mathsf{T}^r, \mathsf{T}^w\rangle$. Si p est dans l'ensemble $\mathcal{P}_{\mathsf{T}_1\sqcap\mathsf{T}_2}$, puisque $\mathcal{R}$ est une relation de sous-typage, on connaît un sous-type de $\mathsf{T}_{\mathsf{T}_1\sqcap\mathsf{T}_2}|_p$, que ce soit un T quelconque ou bien $\mathsf{T}_1|_p$, suivant la variance de la position p. Dans ces deux cas, ce sous-type doit être de la forme $\mathsf{rw}\langle\mathsf{T}'^r, \mathsf{T}'^w\rangle$, avec $\mathsf{T}^w <: \mathsf{T}'^w <: \mathsf{T}'^r <: \mathsf{T}^r$. La transitivité de $<:$ donne alors le résultat attendu. Si la position p n'est pas dans $\mathcal{P}_{\mathsf{T}_1\sqcap\mathsf{T}_2}$, par définition du pré-type $\mathsf{T}_{\mathsf{T}_1\sqcap\mathsf{T}_2}$, il ne peut s'agir que d'une position correspondant à un type directement extrait du type T_1 ou T_2. Dans les deux cas, la bonne formation de ces types nous assure $\mathsf{T}^w <: \mathsf{T}^r$. $\mathsf{T}_{\mathsf{T}_1\sqcap\mathsf{T}_2}$ est bien un type.

Il reste encore à prouver que $\mathsf{T}_{\mathsf{T}_1\sqcap\mathsf{T}_2}$ est bien la borne inférieure des types T_1 et T_2. Prouvons ce résultat en constatant que la relation

$$\begin{array}{rl} \mathcal{R} = & \{\sqcap(\mathsf{T}_1|_p\,, \mathsf{T}_2|_p\,, \mathsf{T}_{\mathsf{T}_1\sqcap\mathsf{T}_2}|_p) \mid\ p \in \mathcal{P}_{\mathsf{T}_1\sqcap\mathsf{T}_2} \text{ est covariante}\} \\ \cup & \{\sqcup(\mathsf{T}_1|_p\,, \mathsf{T}_2|_p\,, \mathsf{T}_{\mathsf{T}_1\sqcap\mathsf{T}_2}|_p) \mid\ p \in \mathcal{P}_{\mathsf{T}_1\sqcap\mathsf{T}_2} \text{ est contravariante}\} \end{array}$$

est un post-point fixe de la fonction $\mathsf{MeetJoin}$. Bien entendu, la définition même de $\mathsf{T}_{\mathsf{T}_1\sqcap\mathsf{T}_2}$ et le fait qu'il s'agit bien d'un type rendent cette vérification tout à fait routinière. Par exemple, si le triplet $\sqcap(\mathsf{T}_1', \mathsf{T}_2', \mathsf{T}_3')$ est dans cette relation, avec $\mathsf{T}_1' = \mathsf{rw}\langle\mathsf{T}_1'^r, \mathsf{T}_1'^w\rangle$, $\mathsf{T}_2' = \mathsf{w}\langle\mathsf{T}_2'^w\rangle$ et qu'il correspond à une position p covariante, $\mathsf{T}_3' = \mathsf{T}_{\mathsf{T}_1\sqcap\mathsf{T}_2}|_p$ doit être de la forme $\mathsf{rw}\langle\mathsf{T}_1'^r, \mathsf{T}_3'^w\rangle$ pour un certain $\mathsf{T}_3'^w$. Or l'appartenance de p à $\mathcal{P}_{\mathsf{T}_1\sqcap\mathsf{T}_2}$ implique également celle de pw car c'est une autre position commune et les types $\mathsf{T}_1'^w$ et $\mathsf{T}_2'^w$ doivent être $\uparrow$-compatibles. Par conséquent $\sqcup(\mathsf{T}_1'^w, \mathsf{T}_2'^w, \mathsf{T}_3'^w)$ est également dans $\mathcal{R}$. De plus $\mathsf{T}_{\mathsf{T}_1\sqcap\mathsf{T}_2}$ est un type donc nous savons $\mathsf{T}_3'^w <: \mathsf{T}_1'^r$. Les deux conditions sont donc remplies pour conclure que $\sqcap(\mathsf{T}_1', \mathsf{T}_2', \mathsf{T}_3')$ est dans $\mathsf{MeetJoin}(\mathcal{R})$, ce qui achève cette preuve[3]. □

Ce théorème permet alors de prouver facilement que les opérateurs $\sqcap$ et $\sqcup$ sont associatifs. Encore une fois la propriété n'est énoncée et n'est prouvée que pour le cas de la borne inférieure, mais le même raisonnement pourrait être suivi pour la borne supérieure.

[3] et le lecteur ?

Proposition 2.18 (Associativité des bornes)**.** *Quels que soient les types* T_1*,* T_2 *et* T_3*,* $(\mathsf{T}_1 \sqcap \mathsf{T}_2) \sqcap \mathsf{T}_3$ *est définie si et seulement si* $\mathsf{T}_1 \sqcap (\mathsf{T}_2 \sqcap \mathsf{T}_3)$ *l'est aussi, auquel cas elles coïncident.*

Démonstration. Si $(\mathsf{T}_1 \sqcap \mathsf{T}_2) \sqcap \mathsf{T}_3$ est définie, cette borne fournit un sous-type commun à T_1, T_2 et T_3, qui entraîne donc, d'après le théorème 2.17, l'existence de $\mathsf{T}_2 \sqcap \mathsf{T}_3$ et par suite de $\mathsf{T}_1 \sqcap (\mathsf{T}_2 \sqcap \mathsf{T}_3)$. La proposition 2.11 permet alors de conclure que ce minorant est un sous-type de $\mathsf{T}_1 \sqcap (\mathsf{T}_2 \sqcap \mathsf{T}_3)$. Le raisonnement symétrique achève de conclure que ces deux types sont égaux. □

Grâce à l'associativité et la commutativité de ces opérateurs la borne inférieure d'un ensemble fini de types sera parfaitement définie. Cette notion s'avérera particulièrement utile pour parler de l'ensemble des types auxquels un nom est connu au sein d'un système, à la section 2.7.

2.6 Théorie des types inductifs

Nous avons étudié jusqu'à maintenant les propriétés des types coinductifs. Mais les types qui apparaissent explicitement dans les processus et systèmes sont bien entendu toujours des types inductifs, afin de pouvoir être décrits de façon finie, avec des récurseurs. Nous allons donc prouver que les propriétés données sur les types coinductifs peuvent être ramenées au niveau des types inductifs en plusieurs étapes : définition de règles inductives pour le sous-typage et les opérateurs $\sqcap$ et $\sqcup$; identification d'une caractéristique-clef des types inductifs ; et enfin preuve de la coïncidence des versions coinductive et inductive du sous-typage et des opérateurs $\sqcap$ et $\sqcup$, en utilisant la caractéristique-clef.

On se donne donc des systèmes de règles d'inférence permettant de prouver le sous-typage et de calculer les bornes inférieures et supérieures directement sur les types inductifs. Ces règles prennent la forme $\Sigma \vdash \mathsf{T}_1 <: \mathsf{T}_2$ ou $\Sigma \vdash \mathsf{T}_1 \sqcap \mathsf{T}_2 = \mathsf{T}_3$. L'intuition de ces jugements est de rajouter une « mémoire », Σ, pour prendre en compte les récurseurs : des jugements déjà prouvés sont accumulés dans cette mémoire pour éviter de boucler dans l'application des règles d'inférence. Ces ajouts dans la mémoire sont opérés par les règles autorisant le dépliage des récurseurs dans les types. À part ces règles de dépliage, les règles correspondent cas par cas aux définitions des fonctions Sub et $\mathsf{MeetJoin}$. Les règles de sous-typage inductif sont données à la figure 2.4, celles de calcul des bornes inférieure et supérieure sont données aux figures 2.5 et 2.6. Concernant les canaux, de nombreuses règles sont nécessaires. Ainsi, figure 2.4, la règle (SR-CANAL) condense tous les cas mais l'hypothèse complète $\Sigma \vdash \mathsf{T}_2^w <: \mathsf{T}_1^w <: \mathsf{T}_1^r <: \mathsf{T}_2^r$ n'est utile que pour déduire $\Sigma \vdash \mathrm{RW}\langle \mathsf{T}_1^r, \mathsf{T}_1^w \rangle <: \mathrm{RW}\langle \mathsf{T}_2^r, \mathsf{T}_2^w \rangle$. Seules les parties utiles de cette prémisse sont requises pour les autres conclusions : par exemple $\Sigma \vdash \mathsf{T}_2^w <: \mathsf{T}_1^w <: \mathsf{T}_1^r$ suffit à déduire $\Sigma \vdash \mathrm{RW}\langle \mathsf{T}_1^r, \mathsf{T}_1^w \rangle <: \mathrm{W}\langle \mathsf{T}_2^w \rangle$. De façon similaire, figures 2.5 et 2.6, seuls quelques cas sont donnés, mais la formulation exacte des cas manquants est très facile à extraire de la définition de $\mathsf{MeetJoin}$.

On a donc deux façons d'obtenir un résultat comme $\mathsf{T}_1 <: \mathsf{T}_2$ quand les types T_1 et T_2 sont inductifs : montrer que $\mathsf{unfold}(\mathsf{T}_1) <: \mathsf{unfold}(\mathsf{T}_2)$ en utilisant la définition coinductive du sous-typage ; ou prouver le résultat $\vdash \mathsf{T}_1 <: \mathsf{T}_2$ directement à l'aide des règles d'inférence pour le sous-typage inductif. Il en va de même pour les bornes supérieures et inférieures. Afin de résoudre ce dilemme, vérifions que ces deux solutions coïncident. Ces preuves de coïncidence reposent

Fig. 2.4 Règles inductives de sous-typage

(SR-AX) $$\Sigma, \mathsf{T}_1 <: \mathsf{T}_2 \vdash \mathsf{T}_1 <: \mathsf{T}_2$$

(SR-LOC) $$\Sigma \vdash \text{LOC} <: \text{LOC}$$

(SR-CANAL) $$\frac{\Sigma \vdash \mathsf{T}_2^w <: \mathsf{T}_1^w <: \mathsf{T}_1^r <: \mathsf{T}_2^r}{\begin{array}{c}\Sigma \vdash \text{RW}\langle \mathsf{T}_1^r, \mathsf{T}_1^w\rangle <: \text{RW}\langle \mathsf{T}_2^r, \mathsf{T}_2^w\rangle \\ \Sigma \vdash \text{RW}\langle \mathsf{T}_1^r, \mathsf{T}_1^w\rangle <: \text{W}\langle \mathsf{T}_2^w\rangle \\ \Sigma \vdash \text{RW}\langle \mathsf{T}_1^r, \mathsf{T}_1^w\rangle <: \text{R}\langle \mathsf{T}_2^r\rangle \\ \Sigma \vdash \text{W}\langle \mathsf{T}_1^w\rangle <: \text{W}\langle \mathsf{T}_2^w\rangle \\ \Sigma \vdash \text{R}\langle \mathsf{T}_1^r\rangle <: \text{R}\langle \mathsf{T}_2^r\rangle\end{array}}$$

(SR-UPLET) $$\frac{\Sigma \vdash \mathsf{T}_i <: \mathsf{T}'_i \text{ pour tout } i}{\Sigma \vdash (\vec{\mathsf{T}}) <: (\vec{\mathsf{T}'})}$$

(SR-DEP) $$\frac{\Sigma \vdash \mathsf{T}_1 <: \mathsf{T}_2}{\Sigma \vdash \sum \vec{x} : \vec{\text{LOC}}.\ \mathsf{T}_1 <: \sum \vec{x} : \vec{\text{LOC}}.\ \mathsf{T}_2}$$

(SR-CANAL-LOC) $$\frac{\Sigma \vdash \mathsf{C}_1 <: \mathsf{C}_2}{\Sigma \vdash \mathsf{C}_{1@s} <: \mathsf{C}_{2@s}}$$

(SR-REC-G) $$\frac{\Sigma, \text{REC}\,\mathbf{Y}.\ \mathsf{T}_1 <: \mathsf{T}_2 \vdash \mathsf{T}_1\{^{\text{REC}\,\mathbf{Y}.\ \mathsf{T}_1}/_{\mathbf{Y}}\} <: \mathsf{T}_2}{\Sigma \vdash \text{REC}\,\mathbf{Y}.\ \mathsf{T}_1 <: \mathsf{T}_2}$$

(SR-REC-D) $$\frac{\Sigma, \mathsf{T}_1 <: \text{REC}\,\mathbf{Y}.\ \mathsf{T}_2 \vdash \mathsf{T}_1 <: \mathsf{T}_2\{^{\text{REC}\,\mathbf{Y}.\ \mathsf{T}_2}/_{\mathbf{Y}}\}}{\Sigma \vdash \mathsf{T}_1 <: \text{REC}\,\mathbf{Y}.\ \mathsf{T}_2}$$

intuitivement sur un dépliage de la preuve inductive pour obtenir une preuve coinductive, et inversement sur un repli de la preuve coinductive en y trouvant les cycles. L'argument autorisant ce repli vient du fait que les types arborescents correspondant à des types inductifs sont réguliers, et que le nombre de sous-arbres différents d'un arbre régulier, même infini, est fini. Cet argument sera appliqué à un ensemble de sous-termes d'un type inductif afin de garantir que la preuve inductive construite par la preuve est effectivement finie.

Définition 2.19 (Sous-termes d'un type inductif)**.** *L'ensemble des* sous-termes *d'un type inductif est* $\mathit{SousTermes}(\mathsf{T})$ *où* $\mathit{SousTermes}$ *est la fonction associant à* T *le plus petit ensemble de types inductifs tel que les équations suivantes soient vérifiées :*

$$\begin{aligned}
\mathit{SousTermes}(\text{R}\langle \mathsf{T}^r\rangle) &= \{\text{R}\langle \mathsf{T}^r\rangle\} \cup \mathit{SousTermes}(\mathsf{T}^r) \\
\mathit{SousTermes}(\text{W}\langle \mathsf{T}^w\rangle) &= \{\text{W}\langle \mathsf{T}^w\rangle\} \cup \mathit{SousTermes}(\mathsf{T}^w) \\
\mathit{SousTermes}(\text{RW}\langle \mathsf{T}^r, \mathsf{T}^w\rangle) &= \{\text{RW}\langle \mathsf{T}^r, \mathsf{T}^w\rangle\} \cup \mathit{SousTermes}(\mathsf{T}^r) \cup \mathit{SousTermes}(\mathsf{T}^w) \\
\mathit{SousTermes}(\text{LOC}) &= \{\text{LOC}\} \\
\mathit{SousTermes}(\mathsf{C}_{@s}) &= \{\mathsf{C}_{@s}\} \cup \mathit{SousTermes}(\mathsf{C}) \\
\mathit{SousTermes}((\mathsf{T}_1, \ldots, \mathsf{T}_n)) &= \{(\mathsf{T}_1, \ldots, \mathsf{T}_n)\} \cup \textstyle\bigcup_i \mathit{SousTermes}(\mathsf{T}_i) \\
\mathit{SousTermes}(\textstyle\sum \vec{x} : \vec{\text{LOC}}.\ \mathsf{T}) &= \{\textstyle\sum \vec{x} : \vec{\text{LOC}}.\ \mathsf{T}\} \cup \mathit{SousTermes}(\mathsf{T}) \\
\mathit{SousTermes}(\text{REC}\,\mathbf{Y}.\ \mathsf{T}) &= \{\text{REC}\,\mathbf{Y}.\ \mathsf{T}\} \cup \mathit{SousTermes}(\mathsf{T}\{^{\text{REC}\,\mathbf{Y}.\ \mathsf{T}}/_{\mathbf{Y}}\})
\end{aligned}$$

La définition du cas récursif, le type $\text{REC}\,\mathbf{Y}.\ \mathsf{T}$, repose directement sur la définition de la fonction sur le type $\mathsf{T}\{^{\text{REC}\,\mathbf{Y}.\ \mathsf{T}}/_{\mathbf{Y}}\}$. Ce choix ne pose pas de

Fig. 2.5 Règles inductives pour $\sqcap$

$$\text{(MEET-AX)}\quad \Sigma, \mathsf{T}_1 \sqcap \mathsf{T}_2 = \mathsf{T}_3 \vdash \mathsf{T}_1 \sqcap \mathsf{T}_2 = \mathsf{T}_3$$

$$\text{(MEET-CANAL-R-R)}\quad \frac{\Sigma \vdash \mathsf{T}_1^r \sqcap \mathsf{T}_2^r = \mathsf{T}_3^r}{\Sigma \vdash \mathrm{R}\langle \mathsf{T}_1^r \rangle \sqcap \mathrm{R}\langle \mathsf{T}_2^r \rangle = \mathrm{R}\langle \mathsf{T}_3^r \rangle}$$

$$\vdots$$

$$\text{(MEET-CANAL-RW-RW)}\quad \frac{\Sigma \vdash \mathsf{T}_1^r \sqcap \mathsf{T}_2^r = \mathsf{T}_3^r \qquad \Sigma \vdash \mathsf{T}_1^w \sqcup \mathsf{T}_2^w = \mathsf{T}_3^w \qquad \vdash \mathsf{T}_3^w <: \mathsf{T}_3^r}{\Sigma \vdash \mathrm{RW}\langle \mathsf{T}_1^r, \mathsf{T}_1^w \rangle \sqcap \mathrm{RW}\langle \mathsf{T}_2^r, \mathsf{T}_2^w \rangle = \mathrm{RW}\langle \mathsf{T}_3^r, \mathsf{T}_3^w \rangle}$$

$$\text{(MEET-LOC)}\quad \Sigma \vdash \mathrm{LOC} \sqcap \mathrm{LOC} = \mathrm{LOC}$$

$$\text{(MEET-UPLET)}\quad \frac{\Sigma \vdash \mathsf{T}_i \sqcap \mathsf{T}'_i = \mathsf{T}''_i}{\Sigma \vdash (\vec{\mathsf{T}}) \sqcap (\vec{\mathsf{T}}') = (\vec{\mathsf{T}}'')}$$

$$\text{(MEET-DEP)}\quad \frac{\Sigma \vdash \mathsf{T}_1 \sqcap \mathsf{T}_2 = \mathsf{T}_3}{\Sigma \vdash \sum \vec{x} : \mathrm{L}\vec{\mathrm{O}}\mathrm{C}.\ \mathsf{T}_1 \sqcap \sum \vec{x} : \mathrm{L}\vec{\mathrm{O}}\mathrm{C}.\ \mathsf{T}_2 = \sum \vec{x} : \mathrm{L}\vec{\mathrm{O}}\mathrm{C}.\ \mathsf{T}_3}$$

$$\text{(MEET-CANAL-LOC)}\quad \frac{\Sigma \vdash \mathsf{C}_1 \sqcap \mathsf{C}_2 = \mathsf{C}_3}{\Sigma \vdash \mathsf{C}_{1@s} \sqcap \mathsf{C}_{2@s} = \mathsf{C}_{3@s}}$$

$$\text{(MEET-REC-1)}\quad \frac{\Sigma, \mathrm{REC}\,\mathbf{Y}.\ \mathsf{T}'_1 \sqcap \mathsf{T}_2 = \mathsf{T}_3 \vdash \mathsf{T}'_1\{^{\mathrm{REC}\,\mathbf{Y}.\ \mathsf{T}'_1}/_{\mathbf{Y}}\} \sqcap \mathsf{T}_2 = \mathsf{T}_3}{\Sigma \vdash \mathrm{REC}\,\mathbf{Y}.\ \mathsf{T}'_1 \sqcap \mathsf{T}_2 = \mathsf{T}_3}$$

$$\text{(MEET-REC-2)}\quad \frac{\Sigma, \mathsf{T}_1 \sqcap \mathrm{REC}\,\mathbf{Y}.\ \mathsf{T}'_2 = \mathsf{T}_3 \vdash \mathsf{T}_1 \sqcap \mathsf{T}'_2\{^{\mathrm{REC}\,\mathbf{Y}.\ \mathsf{T}'_2}/_{\mathbf{Y}}\} = \mathsf{T}_3}{\Sigma \vdash \mathsf{T}_1 \sqcap \mathrm{REC}\,\mathbf{Y}.\ \mathsf{T}'_2 = \mathsf{T}_3}$$

$$\text{(MEET-REC-3)}\quad \frac{\Sigma, \mathsf{T}_1 \sqcap \mathsf{T}_2 = \mathrm{REC}\,\mathbf{Y}.\ \mathsf{T}'_3 \vdash \mathsf{T}_1 \sqcap \mathsf{T}_2 = \mathsf{T}'_3\{^{\mathrm{REC}\,\mathbf{Y}.\ \mathsf{T}'_3}/_{\mathbf{Y}}\}}{\Sigma \vdash \mathsf{T}_1 \sqcap \mathsf{T}_2 = \mathrm{REC}\,\mathbf{Y}.\ \mathsf{T}'_3}$$

problème majeur dans la mesure où l'on donne la définition d'une fonction et non d'un algorithme et que pour tout type T, $\mathsf{SousTermes}(\mathsf{T})$ est défini comme le plus petit point fixe vérifiant les équations de définition de $\mathsf{SousTermes}$. Ainsi, si on considère le type $\mathrm{REC}\,\mathbf{Y}.\ \mathsf{T}$ et que T ne contient pas de récurseur, on se convainc aisément que les équations de définition de $\mathsf{SousTermes}$ imposent que $\mathsf{SousTermes}(\mathrm{REC}\,\mathbf{Y}.\ \mathsf{T})$ soit une union de singletons et, éventuellement, de $\mathsf{SousTermes}(\mathrm{REC}\,\mathbf{Y}.\ \mathsf{T})$. Par exemple,

$$\begin{aligned}&\mathsf{SousTermes}(\mathrm{REC}\,\mathbf{Y}.\ \mathrm{R}\langle\mathbf{Y}\rangle) =\\ &\quad \{\mathrm{REC}\,\mathbf{Y}.\ \mathrm{R}\langle\mathbf{Y}\rangle\} \cup \{\mathrm{R}\langle\mathrm{REC}\,\mathbf{Y}.\ \mathrm{R}\langle\mathbf{Y}\rangle\rangle\} \cup \mathsf{SousTermes}(\mathrm{REC}\,\mathbf{Y}.\ \mathrm{R}\langle\mathbf{Y}\rangle)\end{aligned}$$

Sous cette forme développée, le plus petit ensemble vérifiant l'équation devient évident, $\{\mathrm{REC}\,\mathbf{Y}.\ \mathrm{R}\langle\mathbf{Y}\rangle\} \cup \{\mathrm{R}\langle\mathrm{REC}\,\mathbf{Y}.\ \mathrm{R}\langle\mathbf{Y}\rangle\rangle\}$ dans l'exemple. Par induction dans le cas où T contient d'autres récurseurs, on obtient simplement le fait que l'ensemble des sous-termes de $\mathrm{REC}\,\mathbf{Y}.\ \mathsf{T}$ est fini.

Cette notion permet de prouver les deux propositions suivantes.

Fig. 2.6 Règles inductives pour $\sqcup$

$$\text{(JOIN-AX)}\quad \Sigma, \mathsf{T}_1 \sqcup \mathsf{T}_2 = \mathsf{T}_3 \vdash \mathsf{T}_1 \sqcup \mathsf{T}_2 = \mathsf{T}_3$$

$$\text{(JOIN-CANAL-R-R-R)}\quad \frac{\Sigma \vdash \mathsf{T}_1^r \sqcup \mathsf{T}_2^r = \mathsf{T}_3^r}{\Sigma \vdash \mathrm{R}\langle \mathsf{T}_1^r \rangle \sqcup \mathrm{R}\langle \mathsf{T}_2^r \rangle = \mathrm{R}\langle \mathsf{T}_3^r \rangle}$$

$$\vdots$$

$$\text{(JOIN-CANAL-RW-RW-RW)}\quad \frac{\begin{array}{cc}\Sigma \vdash \mathsf{T}_1^r \sqcup \mathsf{T}_2^r = \mathsf{T}_3^r & \Sigma \vdash \mathsf{T}_1^w \sqcap \mathsf{T}_2^w = \mathsf{T}_3^w \\ \vdash \mathsf{T}_1^w <: \mathsf{T}_1^r & \vdash \mathsf{T}_2^w <: \mathsf{T}_2^r\end{array}}{\Sigma \vdash \mathrm{RW}\langle \mathsf{T}_1^r, \mathsf{T}_1^w \rangle \sqcup \mathrm{RW}\langle \mathsf{T}_2^r, \mathsf{T}_2^w \rangle = \mathrm{RW}\langle \mathsf{T}_3^r, \mathsf{T}_3^w \rangle}\ {}^{\mathsf{T}_1^r \uparrow \mathsf{T}_2^r}\quad {}^{\mathsf{T}_1^w \downarrow \mathsf{T}_2^w}$$

$$\text{(JOIN-LOC)}\quad \Sigma \vdash \mathrm{LOC} \sqcup \mathrm{LOC} = \mathrm{LOC}$$

$$\text{(JOIN-UPLET)}\quad \frac{\Sigma \vdash \mathsf{T}_i \sqcup \mathsf{T}'_i = \mathsf{T}''_i}{\Sigma \vdash (\vec{\mathsf{T}}) \sqcup (\vec{\mathsf{T}}') = (\vec{\mathsf{T}}'')}$$

$$\text{(JOIN-DEP)}\quad \frac{\Sigma \vdash \mathsf{T}_1 \sqcup \mathsf{T}_2 = \mathsf{T}_3}{\Sigma \vdash \sum \vec{x} : \vec{\mathrm{LOC}}.\ \mathsf{T}_1 \sqcup \sum \vec{x} : \vec{\mathrm{LOC}}.\ \mathsf{T}_2 = \sum \vec{x} : \vec{\mathrm{LOC}}.\ \mathsf{T}_3}$$

$$\text{(JOIN-CANAL-LOC)}\quad \frac{\Sigma \vdash \mathsf{C}_1 \sqcup \mathsf{C}_2 = \mathsf{C}_3}{\Sigma \vdash \mathsf{C}_{1@s} \sqcup \mathsf{C}_{2@s} = \mathsf{C}_{3@s}}$$

$$\text{(JOIN-REC-1)}\quad \frac{\Sigma, \mathrm{REC}\,\mathbf{Y}.\ \mathsf{T}'_1 \sqcup \mathsf{T}_2 = \mathsf{T}_3 \vdash \mathsf{T}'_1\{^{\mathrm{REC}\,\mathbf{Y}.\ \mathsf{T}'_1}/_{\mathbf{Y}}\} \sqcup \mathsf{T}_2 = \mathsf{T}_3}{\Sigma \vdash \mathrm{REC}\,\mathbf{Y}.\ \mathsf{T}'_1 \sqcup \mathsf{T}_2 = \mathsf{T}_3}$$

$$\text{(JOIN-REC-2)}\quad \frac{\Sigma, \mathsf{T}_1 \sqcup \mathrm{REC}\,\mathbf{Y}.\ \mathsf{T}'_2 = \mathsf{T}_3 \vdash \mathsf{T}_1 \sqcup \mathsf{T}'_2\{^{\mathrm{REC}\,\mathbf{Y}.\ \mathsf{T}'_2}/_{\mathbf{Y}}\} = \mathsf{T}_3}{\Sigma \vdash \mathsf{T}_1 \sqcup \mathrm{REC}\,\mathbf{Y}.\ \mathsf{T}'_2 = \mathsf{T}_3}$$

$$\text{(JOIN-REC-3)}\quad \frac{\Sigma, \mathsf{T}_1 \sqcup \mathsf{T}_2 = \mathrm{REC}\,\mathbf{Y}.\ \mathsf{T}'_3 \vdash \mathsf{T}_1 \sqcup \mathsf{T}_2 = \mathsf{T}'_3\{^{\mathrm{REC}\,\mathbf{Y}.\ \mathsf{T}'_3}/_{\mathbf{Y}}\}}{\Sigma \vdash \mathsf{T}_1 \sqcup \mathsf{T}_2 = \mathrm{REC}\,\mathbf{Y}.\ \mathsf{T}'_3}$$

Proposition 2.20 (Le sous-typage coinductif sur les types inductifs est le sous-typage inductif)**.** *Quels que soient les types inductifs* T_1 *et* T_2*,*

$$(\mathit{unfold}(\mathsf{T}_1), \mathit{unfold}(\mathsf{T}_2)) \in \nu\mathsf{Sub}$$

si et seulement si $\vdash \mathsf{T}_1 <: \mathsf{T}_2$ *est prouvable.*

La preuve de cette proposition est très similaire à la preuve de la suivante, elle est donc laissée au lecteur.

Proposition 2.21 (Les bornes inférieures et supérieures coinductives sur des types inductifs sont les bornes inductives)**.** *Quels que soient les types inductifs* T_1*,* T_2 *et* T_3*,* $\sqcap(\mathit{unfold}(\mathsf{T}_1), \mathit{unfold}(\mathsf{T}_2), \mathit{unfold}(\mathsf{T}_3)) \in \nu\mathsf{MeetJoin}$ *si et seulement si* $\vdash \mathsf{T}_1 \sqcap \mathsf{T}_2 = \mathsf{T}_3$ *est prouvable. Il en va de même pour* $\sqcup$*.*

Démonstration. Commençons par prouver que l'on peut extraire une preuve de $\vdash \mathsf{T}_1 \sqcap \mathsf{T}_2 = \mathsf{T}_3$ de $\sqcap(\mathsf{unfold}(\mathsf{T}_1), \mathsf{unfold}(\mathsf{T}_2), \mathsf{unfold}(\mathsf{T}_3)) \in \nu\mathsf{MeetJoin}$, et son dual pour $\sqcup$. Considérons pour cela des types T_1, T_2, et T_3 et un jeu d'hypothèses Σ'

de la forme $\mathsf{T}'_1 \sqcap \mathsf{T}'_2 = \mathsf{T}'_3$ ou $\mathsf{T}'_1 \sqcup \mathsf{T}'_2 = \mathsf{T}'_3$ où T'_i est un sous-terme de T_i et où $\sqcap(\mathsf{unfold}(\mathsf{T}'_1), \mathsf{unfold}(\mathsf{T}'_2), \mathsf{unfold}(\mathsf{T}'_3))$ ou $\sqcup(\mathsf{unfold}(\mathsf{T}'_1), \mathsf{unfold}(\mathsf{T}'_2), \mathsf{unfold}(\mathsf{T}'_3))$, suivant le cas, est dans $\nu\mathsf{MeetJoin}$.

Prouvons que $\sqcap(\mathsf{unfold}(\mathsf{T}'_1)\mathsf{unfold}(\mathsf{T}'_2), \mathsf{unfold}(\mathsf{T}'_3)) \in \nu\mathsf{MeetJoin}$ implique que $\Sigma' \vdash \mathsf{T}'_1 \sqcap \mathsf{T}'_2 = \mathsf{T}'_3$. Construisons donc la preuve en raisonnant suivant la forme des types T'_i. L'ordre bien fondé assurant la terminaison de cette construction est l'ordre lexicographique $(|\mathsf{SousTermes}\mathsf{T}_1| \cdot |\mathsf{SousTermes}\mathsf{T}_2| \cdot |\mathsf{SousTermes}\mathsf{T}_3| - |\Sigma'|, |\mathsf{T}'_1| + |\mathsf{T}'_2| + |\mathsf{T}'_3|)$ où $|\mathsf{SousTermes}\mathsf{T}_i|$ est le cardinal de l'ensemble des sous-termes de T_i et $|\mathsf{T}'_1|$ la taille du type T'_1.

- Supposons que l'un des T'_i soit de la forme $\textsc{rec}\,\mathbf{Y}.\,\mathsf{T}$. Si $\mathsf{T}'_1 \sqcap \mathsf{T}'_2 = \mathsf{T}'_3$ est une hypothèse de Σ', la preuve peut être conclue directement par l'axiome (MEET-AX). Sinon, la règle (MEET-REC-i) peut être appliquée. La mémoire Σ' est alors remplacée par $\Sigma', \mathsf{T}'_1 \sqcap \mathsf{T}'_2 = \mathsf{T}'_3$ avec $|\Sigma', \mathsf{T}'_1 \sqcap \mathsf{T}'_2 = \mathsf{T}'_3| > |\Sigma'|$, ce qui garantit la décroissance stricte suivant l'ordre choisi. On peut donc conclure avec l'hypothèse d'induction.
- Si aucun des T'_i n'est de la forme $\textsc{rec}\,\mathbf{Y}.\,\mathsf{T}$, la définition de $\mathsf{unfold}(\mathsf{T}'_i)$ implique que $\mathsf{unfold}(\mathsf{T}'_i)$ et T'_i partagent le même constructeur de tête. Quel que soit le cas de la définition de $\mathsf{MeetJoin}$ pour le triplet

  $$\sqcap(\mathsf{unfold}(\mathsf{T}'_1), \mathsf{unfold}(\mathsf{T}'_2), \mathsf{unfold}(\mathsf{T}'_3))$$

  on utilise alors la règle qui correspond pour $\Sigma' \vdash \mathsf{T}'_1 \sqcap \mathsf{T}'_2 = \mathsf{T}'_3$. Regardons un cas typique :
  - Si tous les T'_i sont de la forme $\mathrm{R}\langle \mathsf{T}''_i \rangle$, par définition de $\mathsf{MeetJoin}$, on sait que $\sqcap(\mathsf{unfold}(\mathsf{T}''_1), \mathsf{unfold}(\mathsf{T}''_2), \mathsf{unfold}(\mathsf{T}''_3))$ doit être dans $\nu\mathsf{MeetJoin}$. $|\mathsf{T}''_i| < |\mathsf{T}'_i|$, on peut donc utiliser l'hypothèse d'induction pour conclure $\Sigma' \vdash \mathsf{T}''_1 \sqcap \mathsf{T}''_2 = \mathsf{T}''_3$ d'où les règles (MEET-CANAL-$\cdots$) donnent $\Sigma' \vdash \mathsf{T}'_1 \sqcap \mathsf{T}'_2 = \mathsf{T}'_3$.

Le raisonnement est similaire pour $\sqcup$.

Réciproquement, considérons une preuve de $\vdash \mathsf{T}_1 \sqcap \mathsf{T}_2 = \mathsf{T}_3$ et la relation

$$\begin{aligned}
\mathcal{R} = \quad & \{\sqcap(\mathsf{unfold}(\mathsf{T}'_1), \mathsf{unfold}(\mathsf{T}'_2), \mathsf{unfold}(\mathsf{T}'_3)) \\
& \quad | \; \exists \Sigma' \text{ tel que } \Sigma' \vdash \mathsf{T}'_1 \sqcap \mathsf{T}'_2 = \mathsf{T}'_3 \\
& \qquad \text{apparaît dans la preuve de } \vdash \mathsf{T}_1 \sqcap \mathsf{T}_2 = \mathsf{T}_3\} \\
\cup \quad & \{\sqcup(\mathsf{unfold}(\mathsf{T}'_1), \mathsf{unfold}(\mathsf{T}'_2), \mathsf{unfold}(\mathsf{T}'_3)) \\
& \quad | \; \exists \Sigma' \text{ tel que } \Sigma' \vdash \mathsf{T}'_1 \sqcup \mathsf{T}'_2 = \mathsf{T}'_3 \\
& \qquad \text{apparaît dans la preuve de } \vdash \mathsf{T}_1 \sqcap \mathsf{T}_2 = \mathsf{T}_3\}
\end{aligned}$$

Prouvons que $\mathcal{R}$ est un post-point fixe de $\mathsf{MeetJoin}$. Considérons pour cela un triplet de $\mathcal{R}$ et raisonnons sur la dernière règle utilisée pour obtenir le jugement correspondant dans la preuve de $\vdash \mathsf{T}_1 \sqcap \mathsf{T}_2 = \mathsf{T}_3$.

(meet-ax) $\mathsf{T}'_1 \sqcap \mathsf{T}'_2 = \mathsf{T}'_3$ doit alors avoir été introduit dans Σ' par une application préalable d'une des règles (MEET-REC-i) dans la même branche de la preuve. Considérons la première règle au-dessus de cette règle (MEET-REC-i) qui ne soit pas elle-même une règle (MEET-REC-j). Puisque les types sont contractifs, on sait qu'il en existe une avant la règle axiome. Quelle que soit cette règle, on sait aussi que sa conclusion est de la forme $\mathsf{T}''_1 \sqcap \mathsf{T}''_2 = \mathsf{T}''_3$ où $\mathsf{unfold}(\mathsf{T}'_k) = \mathsf{unfold}(\mathsf{T}''_k)$, pour k dans $\{1, 2, 3\}$, car la définition de $\mathsf{unfold}(\cdot)$ impose $\mathsf{unfold}(\textsc{rec}\,\mathbf{Y}.\,\mathsf{T}) = \mathsf{unfold}(\mathsf{T}\{^{\textsc{rec}\,\mathbf{Y}.\,\mathsf{T}}/_{\mathbf{Y}}\})$. On peut donc appliquer le même raisonnement que dans le cas de cette règle pour prouver que $\sqcap(\mathsf{unfold}(\mathsf{T}'_1), \mathsf{unfold}(\mathsf{T}'_2), \mathsf{unfold}(\mathsf{T}'_3))$ doit être dans $\mathcal{R}$.

(meet-rec-i) Le raisonnement est très similaire au cas précédent : la contractivité des types assure en effet qu'une règle distincte des (MEET-AX) et (MEET-REC-j) soit utilisée au-dessus de la série de (MEET-REC-j) à laquelle (MEET-REC-i) appartient. Puisque les règles (MEET-REC-j) ne modifient pas les types coinductifs associés aux types inductifs apparaissant dans le jugement, la « preuve coinductive », c'est-à-dire la déconstruction d'un constructeur de type, peut procéder en suivant le schéma de la première règle non (MEET-REC-j) au-dessus de (MEET-REC-i).

(meet-canal-r-r) Si $\mathsf{T}'_1 \sqcap \mathsf{T}'_2 = \mathsf{T}'_3$ est de la forme $\mathrm{R}\langle \mathsf{T}''_1 \rangle \sqcap \mathrm{R}\langle \mathsf{T}''_2 \rangle = \mathrm{R}\langle \mathsf{T}''_3 \rangle$, la preuve de $\Sigma' \vdash \mathsf{T}'_1 \sqcap \mathsf{T}'_2 = \mathsf{T}'_3$ contient forcément une preuve de $\Sigma' \vdash \mathsf{T}''_1 \sqcap \mathsf{T}''_2 = \mathsf{T}''_3$. Le triplet $\sqcap(\mathsf{unfold}(\mathsf{T}''_1), \mathsf{unfold}(\mathsf{T}''_2), \mathsf{unfold}(\mathsf{T}''_3))$ de $\mathcal{R}$ permet alors de conclure que $\sqcap(\mathsf{unfold}(\mathsf{T}'_1), \mathsf{unfold}(\mathsf{T}'_2), \mathsf{unfold}(\mathsf{T}'_3))$ est dans $\mathsf{MeetJoin}(\mathcal{R})$.

Tous les autres cas sont similaires.

□

La coïncidence des deux théories, l'inductive et la coinductive, permet donc de choisir la version la plus intuitive pour chaque preuve. Ainsi, pour de nombreuses preuves comme celles établissant que l'opérateur $\sqcap$ calcule effectivement la borne inférieure dans l'ensemble des types ordonné par le sous-typage, l'approche coinductive permet de procéder suivant les intuitions habituelles sur les inductifs, c'est-à-dire par cas suivant la forme des types, sans devoir prendre en compte les récurseurs. L'absence des récurseurs permet aussi d'utiliser directement l'égalité extensionnelle sur les types au lieu d'utiliser une fastidieuse équivalence.

Maintenant que la théorie des types associés aux noms et variables est cohérente et bien établie, intéressons-nous au typage des processus et systèmes qu'elle autorise.

2.7 Typage des systèmes et processus

Maintenant que nous avons développé une théorie riche pour les types de valeurs, revenons sur les objectifs essentiels du typage. En effet, le typage que nous avons commencé à mettre en place est un outil d'analyse des processus et des systèmes permettant de détecter des erreurs qui s'avéreraient fatales au cours de leur exécution et surtout d'assurer que les processus n'utilisent jamais de capacités qui ne leur ont pas été communiquées.

Nous allons plus particulièrement nous intéresser ici au typage statique, c'est-à-dire un typage effectué avant que l'exécution proprement dite ne soit entamée. L'analyse des processus et systèmes consistera donc à attribuer à tout identifiant un des types présentés auparavant et à vérifier si un processus ou un système utilise effectivement ses identifiants selon les types qui leur sont associés. Cette analyse permettra d'éviter tout à la fois les comportements illicites et erronés, puisque dans les deux cas le problème provient d'un usage non conforme des identifiants. Contrairement aux noms et variables pour lesquels il existe toute une palette de types distincts, les processus et systèmes seront simplement classés en deux catégories : les bien et les mal formés.

On attribuera un type E à un identifiant s par des *hypothèses* de la forme $s : \mathsf{E}$. On appellera *environnements* des listes de telles hypothèses que l'on notera souvent Γ, Φ ou Ω. $\Gamma(s)$ désignera l'ensemble des types associés à l'identifiant s

Fig. 2.7 Environnements de typage

$$\text{(E-VIDE)} \qquad \vdash \mathbf{env}$$

$$\text{(E-LOC)} \qquad \frac{\Gamma \vdash \mathbf{env}}{\Gamma, s : \text{LOC} \vdash \mathbf{env}} \downarrow(\Gamma(s) \cup \{\text{LOC}\})$$

$$\text{(E-CANAL)} \qquad \frac{\Gamma \vdash \mathbf{env} \quad \Gamma \vdash s : \text{LOC}}{\Gamma, u : \mathsf{C}@s \vdash \mathbf{env}} \downarrow(\Gamma(u) \cup \{\mathsf{C}@s\})$$

par des hypothèses de l'environnement Γ et le *domaine* de Γ, noté $\mathsf{dom}(\Gamma)$ sera l'ensemble des identifiants auxquels Γ associe un type. Enfin, dans la mesure où la primitive new du langage associe à un nom a un type de la forme E, cela pourra être vu comme une hypothèse de la forme $a : \mathsf{E}$; on emploiera donc éventuellement la notation générale $(\mathsf{new}\,\Phi)$ pour désigner la cascade de primitives $(\mathsf{new}\,a_1 : \mathsf{E}_1)\,(\mathsf{new}\,a_2 : \mathsf{E}_2)\ldots$ quand Φ est l'environnement $a_1 : \mathsf{E}_1; a_2 : \mathsf{E}_2; \ldots$

Il est cependant indispensable de poser des contraintes sur les hypothèses qui constituent un environnement. Par exemple si un environnement contenait à la fois les deux hypothèses $a : \text{RW}\langle\text{R}\langle\rangle\rangle@l$ et $a : \text{RW}\langle\rangle@l$, la vérification qu'un processus utilise effectivement a au type auquel il est associé n'aurait guère de sens. Puisque les environnements sont des listes, on pourra garantir qu'ils ne contiennent pas d'hypothèses contradictoires en vérifiant que chaque hypothèse rajoutée à la liste est cohérente avec l'ensemble des hypothèses préalables portant sur le même identifiant. D'ailleurs, la théorie des types permet de donner un critère de cohérence simple : on imposera que les types associés à un même identifiant soient tous compatibles.

On exprimera le fait qu'un environnement Γ est bien formé par un jugement $\Gamma \vdash \mathbf{env}$. Les règles d'inférence pour ces jugements sont données figure 2.7. Ces règles restent très simples :

- la première indique que l'environnement vide est bien formé ;
- la deuxième affirme que l'on peut rajouter une hypothèse s : LOC à un environnement Γ bien formé dès que l'identifiant s est inconnu ou déjà associé au type LOC dans Γ. Vérifier que les types $\Gamma(s)$ sont compatibles avec $\{\text{LOC}\}$ revient en effet à imposer que $\Gamma(s)$ soit $\emptyset$ ou $\{\text{LOC}\}$;
- la troisième effectue le même raisonnement pour les identifiants de canaux.

La différence de statuts entre les identifiants normaux u et les identifiants étendus s sera expliquée avec le typage des processus récursifs, car seule cette règle de typage introduit des variables de récursion dans l'environnement.

Bien entendu, ces environnements seront utilisés pour les jugements portant sur les systèmes et processus. Par exemple, le classement d'un système M dans la catégorie « bien formés » sous l'ensemble d'hypothèses Γ sera décrit par le jugement $\Gamma \vdash M$. Concernant un processus P, la vérification qu'il est bien formé devra prendre en compte la localité dans laquelle le processus s'exécute. Par exemple, pour pouvoir typer le simple processus $\mathsf{goto}\,l.\ a\,!\,\langle\rangle$ sous les hypothèses Γ, il faudra préalablement vérifier que l'environnement Γ associe effectivement au nom a un canal de la localité l, c'est-à-dire la localité dans laquelle $a\,!\,\langle\rangle$ s'exécutera. Il faudra aussi vérifier que l'émission de messages sur ce canal est

Fig. 2.8 Typage des valeurs

$$\text{(V-ID)}\quad \frac{\Gamma, s : \mathsf{E}, \Gamma' \vdash \mathbf{env}}{\Gamma, s : \mathsf{E}, \Gamma' \vdash s : \mathsf{E}} \qquad \frac{\Gamma \vdash s : \mathsf{E}_1 \quad \Gamma \vdash s : \mathsf{E}_2}{\Gamma \vdash s : \mathsf{E}_3}\ \mathsf{E}_1 \sqcap \mathsf{E}_2 <: \mathsf{E}_3 \quad \text{(V-INF)}$$

$$\text{(V-LOCALISATION)}\quad \frac{\Gamma \vdash s : \text{LOC} \quad \Gamma \vdash t : \mathsf{E}}{\Gamma \vdash_s t : \mathsf{E}} \qquad \frac{\Gamma \vdash s : \text{LOC} \quad \Gamma \vdash u : \mathsf{C}@s}{\Gamma \vdash_s u : \mathsf{C}} \quad \text{(V-CANAL-LOC)}$$

$$\text{(V-UPLET)}\quad \frac{\Gamma \vdash_s V_i : \mathsf{T}_i}{\Gamma \vdash_s (\vec{V}) : (\vec{\mathsf{T}})} \qquad \frac{\Gamma \vdash_s t_i : \text{LOC} \quad \Gamma \vdash_s V : \mathsf{T}\{\vec{t}/\vec{x}\}}{\Gamma \vdash_s ((\vec{t}), V) : \sum \vec{x} : \vec{\text{LOC}}.\ \mathsf{T}} \quad \text{(V-DEP)}$$

autorisée, c'est-à-dire que l'environnement Γ associe à a le type $\text{W}\langle\rangle@l$ ou un de ses sous-types. On notera alors $\Gamma \vdash_l a : \text{W}\langle\rangle$ le jugement correspondant à cette vérification. Les règles du sous-typage seront prises en compte afin que ce jugement soit vrai dans l'environnement $\Gamma = l : \text{LOC}, a : \text{RW}\langle\rangle@l$.

Commençons donc par décrire les jugements portant sur les valeurs. Les règles d'inférence pour le typage des valeurs sont données à la figure 2.8. La règle (V-INF) donne tout son sens à la notion de borne inférieure, en explicitant comment elle est utilisée pour combiner en un seul type les diverses capacités connues pour un identifiant. Ces règles se répartissent en deux catégories distinctes : les règles qui permettent le typage des valeurs, qui sont localisées dans la mesure où leur conclusion $\cdots \vdash_s \cdots$ mentionne explicitement la localité où le jugement est valable, et les règles qui ne permettent que le typage des identifiants, en ne procédant qu'à une simple extraction de l'information contenue dans l'environnement. Les secondes sont nécessaires pour garantir que s est une localité quand il apparaît dans un jugement $\cdots \vdash_s \cdots$. Les premières s'avéreront utiles au typage des processus, puisque les jugements correspondants seront eux aussi localisés. La différence entre les règles (V-LOCALISATION) et (V-CANAL-LOC) met en lumière l'information que porte la localisation. Par exemple, dans l'environnement $\Gamma = l : \text{LOC}, k : \text{LOC}, a : \text{RW}\langle\rangle@l$, les jugements $\Gamma \vdash_l a : \text{RW}\langle\rangle@l$, $\Gamma \vdash_k a : \text{RW}\langle\rangle@l$ et $\Gamma \vdash_l a : \text{RW}\langle\rangle$ sont prouvables. En revanche, $\Gamma \vdash_k a : \text{RW}\langle\rangle$ ne le sera pas. En effet, le Dπ-calcul laisse une place prépondérante aux ressources locales : les communications ne peuvent avoir lieu que sur des canaux locaux, il est donc particulièrement intéressant de tracer un fossé les distinguant en ajoutant des types spécifiques pour les canaux locaux.

Cette différence de traitements transparaît aussi dans les types que peuvent avoir les valeurs transmises lors de communications : ils doivent être clos. Cela signifie que la transmission d'un canal c de type $\text{RW}\langle\rangle@l$ ne peut se faire que de deux façons différentes. La première façon est de transmettre ce canal au type simple $\text{RW}\langle\rangle$:

$$l[\![a\,!\,\langle c\rangle\, P]\!] \mid l[\![a\,?\,(x : \text{RW}\langle\rangle)\, Q]\!]$$

Sa transmission à ce type ne peut cependant être effectuée qu'au sein de sa localité l. La seule autre façon possible pour le transmettre est la suivante :

$$k[\![b\,!\,\langle(l,c)\rangle\,P']\!] \mid k[\![b\,?\left((x,y) : \sum z : \text{LOC. } \text{RW}\langle\rangle_@ z\right) Q']\!]$$

Puisque l'on veut ici transmettre c ailleurs que dans sa localité l, il est indispensable de le communiquer au type $\text{RW}\langle\rangle_@ l$. La contrainte de clôture des types impose alors d'utiliser une somme dépendante pour fermer la portée de la localité de c. On pourra par exemple utiliser le type clos $\sum z : \text{LOC. } \text{RW}\langle\rangle_@ z$. La règle de typage (V-DEP) indique que les valeurs de ce type sont des couples composés d'une localité et d'un canal. La preuve du jugement $\Gamma \vdash_k (l,c) : \sum z : \text{LOC. } \text{RW}\langle\rangle_@ z$ passe alors par les propriétés $\Gamma \vdash_k l : \text{LOC}$ et $\Gamma \vdash_k c : \text{RW}\langle\rangle_@ l$ car l remplace z comme localité du canal.

Regardons maintenant comment analyser le comportement des processus pour y détecter des anomalies qui pourraient se révéler dangereuses à l'exécution. Comme nous l'avons déjà indiqué, cette vérification tiendra compte de la localité dans laquelle le processus s'exécute. Les règles de typage pour les valeurs, qui sont localisées elles aussi, permettent de considérer sereinement les primitives de communication : le processus $a\,!\,\langle V\rangle\,P$ sera essentiellement bien formé si la valeur V à transmettre a effectivement le type des valeurs transitant sur le canal a, et que sa continuation P est elle-même bien formée. La primitive de réception $a\,?\,(X : \mathsf{T})\,P$ est un peu plus subtile dans la mesure où la vérification de la continuation P devra prendre en compte les connaissances acquises lors de la réception des données X. Cela reviendra simplement à enrichir l'environnement de typage.

Le typage des processus récursifs suppose quant à lui une analyse particulièrement fine. Pour que le processus $(\mathsf{rec}\ Z : \mathsf{R}\,(X)\,.\,P)\,\langle V\rangle$ soit bien typé dans la localité l, il nous faut garantir que le déclenchement du processus P dans l ne provoque aucune erreur, mais aussi que ce déclenchement soit sûr dans toutes les autres localités où pourraient avoir lieu des appels récursifs $Z\,\langle V'\rangle$. Pour prendre un exemple, si le système $l[\![(\mathsf{rec}\ Z : \mathsf{R}\,()\,.\,a\,!\,\langle\rangle\,\mathsf{goto}\,k.\ Z\,\langle\rangle)\,\langle\rangle]\!]$ était typé en se contentant de vérifier que le canal a existe dans la localité l, dès la seconde itération ce système provoquerait une erreur en tentant de communiquer sur le canal a au sein de la localité k. Pour éviter ces problèmes, on se propose de typer le processus en considérant qu'il s'exécute dans une localité neutre, dont les seules propriétés connues seront dictées par le type R de cette récursion. Afin de garantir la neutralité de cette localité, on utilisera simplement la variable de récursion Z elle-même ; les règles de portées des variables empêcheront automatiquement les collisions avec un identifiant de localité existant. Cette astuce justifie l'existence dans la syntaxe d'identifiants ainsi que valeurs étendus, et leur omniprésence dans les règles de typage. Les variables de récursion sont en effet utilisées comme des localités particulières au typage, et les règles de typage des valeurs utilisent donc toujours des identifiants étendus lorsqu'il peut s'agir d'une variable de récursion. Par ailleurs, leur absence de la syntaxe des processus et systèmes donnée figure 1.2 montre qu'il ne s'agit là que d'un artifice de la technique de typage.

Pour que ce choix de typer la partie P du processus $(\mathsf{rec}\ Z : \mathsf{R}\,(V)\,.\,P)\,\langle X\rangle$ dans la « localité » Z tienne la route, il est aussi indispensable de fournir un moyen d'indiquer quelles sont les ressources disponibles dans cette localité. Afin de garantir simplement que ces ressources seront disponibles dans toutes les

localités où le processus pourra être exécuté, la seule possibilité offerte pour les déclarer sera de les passer dans les paramètres du processus récursif. Dans l'exemple précédent, pour communiquer sur le canal a de la localité, le processus pourrait donc être écrit $l[\![(\mathsf{rec}\ Z : \mathsf{R}(x)\,.\ x\,!\,\langle\rangle\ \mathsf{goto}\ k.\ Z\,\langle a_k\rangle)\,\langle a_l\rangle]\!]$, ce qui force bien à identifier deux canaux différents, l'un dans l, l'autre dans k, sur lesquels procéder à la communication. Le type associé au processus récursif, R, découle directement de cette analyse : ce type doit permettre de typer le paramètre x en le laissant dépendre de la localité où il existe, formellement Z, tout comme a_k est lié à k et a_l à l. Dans cet exemple, R sera donc une somme dépendante comme $\sum y : \textsc{loc}.\ \textsc{w}\langle\rangle_@ y$. Pour vérifier que P est bien formé, on pourra alors le supposer s'exécutant dans la localité Z paramétrée par x de sorte que (Z, x) ait pour type R, et les invocations du processus récursifs reposeront sur la vérification que (l, a_l) et (k, a_k) ont également ce type R. Par ailleurs, de façon similaire aux types des valeurs communiquées sur les canaux, on imposera dans la suite que les types des paramètres et des arguments des processus récursifs soient clos, c'est-à-dire que $\mathsf{fn}(\mathsf{R})$ soit vide quel que soit le type de récursion R.

Cette approche laisse une difficulté non résolue : si le typage de P est effectué dans un environnement uniquement enrichi de la connaissance $(Z, x) : \mathsf{R}$, cette connaissance sera ramenée au format normal d'un environnement, c'est-à-dire une liste d'hypothèses associant un type $\textsc{loc}$ ou $\mathsf{C}_@ s$ à chacun des identifiants (ici $Z : \textsc{loc}$ et $x : \textsc{w}\langle\rangle_@ Z$). Or, pour définir le typage par un jeu de règles d'inférences, il est indispensable de conserver dans les jugements le type R intact jusqu'aux appels récursifs $Z\,\langle a_k\rangle$. Le typage des processus se fera donc dans un environnement Γ assisté d'une liste associant un type de la forme R à des variables de récursion, liste qui sera souvent notée Δ. Les règles de typage données à la figure 2.9 portent donc sur des jugements de la forme $\Gamma \mid \Delta \vdash_s P$.

Parmi ces règles de typage, regardons tout d'abord les deux règles les plus emblématiques, concernant le typage de l'émission et de la réception. La règle (T-R) permettant de typer les réceptions prend la forme

$$(\text{T-R})\quad \frac{\Gamma \vdash_s u : \textsc{r}\langle \mathsf{T}\rangle \qquad \Gamma;\langle X : \mathsf{T}\rangle_@ s \mid \Delta \vdash_s P}{\Gamma \mid \Delta \vdash_s u\,?\,(X : \mathsf{T})\,P}$$

L'environnement $\Gamma;\langle X : \mathsf{T}\rangle_@ s$, dans lequel la continuation est typée, utilise plusieurs notations encore non expliquées. Tout d'abord le point-virgule sert à composer les deux environnements Γ et $\langle X : \mathsf{T}\rangle_@ s$ en assurant que les domaines de ces deux environnements ne se chevauchent pas. Cette notation n'impose pas de contrainte forte sur les environnements mais reprend simplement la convention de Barendregt en assurant que les variables liées qui sont rajoutées à l'environnement sont bien distinctes de celles qui y sont déjà définies. Par ailleurs la notation $\langle X : \mathsf{T}\rangle_@ s$ permet de désigner simplement l'environnement dans lequel le type T est associé au motif X en sachant que la localité dans laquelle cette communication va être exécutée est s. Cette expansion de $X : \mathsf{T}$ dans s pour obtenir un environnement est définie formellement à la figure 2.10. L'indication de la localité où l'expansion prend place, en l'occurrence s, sert dans le cas où les valeurs échangées sont des canaux locaux de type C : l'expansion forme dans ce cas le type $\mathsf{C}_@ s$ qui devra apparaître dans l'environnement tout en permettant des échanges très simplifiés de canaux locaux, comme nous l'avions déjà vu à propos du typage des valeurs. Mis à part la vérification du typage de la

Fig. 2.9 Typage des processus

$$\text{(T-W)}\quad \frac{\Gamma \vdash_s u : \mathrm{W}\langle \mathsf{T} \rangle \quad \Gamma \vdash_s V : \mathsf{T} \quad \Gamma \mid \Delta \vdash_s P}{\Gamma \mid \Delta \vdash_s u\,!\,\langle V \rangle\, P}$$

$$\text{(T-R)}\quad \frac{\Gamma \vdash_s u : \mathrm{R}\langle \mathsf{T} \rangle \quad \Gamma; \langle X : \mathsf{T} \rangle_{@s} \mid \Delta \vdash_s P}{\Gamma \mid \Delta \vdash_s u\,?\,(X : \mathsf{T})\, P}$$

$$\text{(T-GOTO)}\quad \frac{\Gamma \vdash u : \mathrm{LOC} \quad \Gamma \mid \Delta \vdash_u P}{\Gamma \mid \Delta \vdash_s \mathsf{goto}\, u.\ P}$$

$$\text{(T-IF)}\quad \frac{[\Gamma]_{u_1=u_2} \mid \Delta \vdash_s P_1 \quad \text{quand } [\Gamma]_{u_1=u_2} \vdash \mathbf{env} \quad \Gamma \mid \Delta \vdash_s P_2}{\Gamma \mid \Delta \vdash_s \mathsf{if}\ u_1 = u_2\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2}$$

$$\text{(T-NEWCHAN)}\quad \frac{\Gamma; c : \mathsf{C}_{@s} \mid \Delta \vdash_s P}{\Gamma \mid \Delta \vdash_s \mathsf{newchan}\, c : \mathsf{C}\, \mathsf{in}\, P}$$

$$\text{(T-NEWLOC)}\quad \frac{\Gamma; \left\langle (k, (\vec{c})) : \sum x : \mathrm{LOC}.\ \vec{\mathsf{C}}_{@x} \right\rangle \mid \Delta \vdash_k P_k \quad \Gamma; \left\langle (k, (\vec{c})) : \sum x : \mathrm{LOC}.\ \vec{\mathsf{C}}_{@x} \right\rangle \mid \Delta \vdash_s P}{\Gamma \mid \Delta \vdash_s \mathsf{newloc}\, k, (\vec{c}) : \sum x : \mathrm{LOC}.\ \vec{\mathsf{C}}_{@x}\, \mathsf{with}\, P_k\, \mathsf{in}\, P}$$

$$\text{(T-PAR)}\quad \frac{\Gamma \mid \Delta \vdash_s P_1 \quad \Gamma \mid \Delta \vdash_s P_2}{\Gamma \mid \Delta \vdash_s P_1 \mid P_2}$$

$$\text{(T-HERE)}\quad \frac{[\Gamma]_{x=s} \mid \Delta \vdash_s P}{\Gamma \mid \Delta \vdash_s \mathsf{here}\,(x)\, P}$$

$$\text{(T-RÉP)}\quad \frac{\Gamma \mid \Delta \vdash_s P}{\Gamma \mid \Delta \vdash_s *P}$$

$$\text{(T-REC)}\quad \frac{\Gamma \vdash_s (s, V) : \mathsf{R} \quad \Gamma; \langle (Z, X) : \mathsf{R} \rangle_{@Z} \mid \Delta; Z : \mathsf{R} \vdash_Z P}{\Gamma \mid \Delta \vdash_s (\mathsf{rec}\ Z : \mathsf{R}\,(X)\,.\ P)\,\langle V \rangle}$$

$$\text{(T-CALL)}\quad \frac{\Gamma \vdash_s (s, V) : \mathsf{R}}{\Gamma \mid \Delta, Z : \mathsf{R}, \Delta' \vdash_s Z\,\langle V \rangle}$$

$$\text{(T-STOP)}\quad \frac{\Gamma \vdash \mathbf{env}}{\Gamma \mid \Delta \vdash_s \mathsf{stop}}$$

Fig. 2.10 Expansion des valeurs

$$\langle u : \mathsf{C}\rangle \,@t = u : \mathsf{C}@t$$
$$\langle u : \mathsf{C}@t_0\rangle \,@t = u : \mathsf{C}@t_0$$
$$\langle s : \textsc{loc}\rangle \,@t = s : \textsc{loc}$$
$$\left\langle ((\vec{s}), S) : \sum \vec{x} : \vec{\textsc{loc}}.\ \mathsf{T}\right\rangle @t = s_1 : \textsc{loc}, \ldots, \left\langle S : \mathsf{T}\{\vec{s}/\vec{x}\}\right\rangle @t$$
$$\langle (S_1, \ldots, S_n) : (\mathsf{T}_1, \ldots, \mathsf{T}_n)\rangle \,@t = \langle S_1 : \mathsf{T}_1\rangle \,@t, \ldots, \langle S_n : \mathsf{T}_n\rangle \,@t$$

continuation, (T-R) assure que la capacité de réception est réellement disponible pour le canal sur lequel elle est effectuée. De la même façon, la règle de typage des émissions (T-W) vérifie simplement que l'environnement de typage dispose de la capacité d'émission pour le canal par le jugement $\Gamma \vdash_s u : \textsc{w}\langle \mathsf{T}\rangle$ et qu'il dispose aussi des capacités émises, c'est-à-dire que la valeur envoyée a effectivement le type exigé par le canal u. En effet, $\Gamma \vdash_s V : \mathsf{T}$ garantit que le jugement

$$l : \textsc{loc}, a : \textsc{w}\langle \textsc{rw}\langle\rangle\rangle @l, c : \textsc{r}\langle\rangle @l \vdash_l a\,!\,\langle c\rangle$$

ne sera pas prouvable : le processus $a\,!\,\langle c\rangle$ y tente de transmettre les capacités d'émission et de réception sur le canal c alors qu'il ne dispose lui-même que de la réception.

La règle (T-REC) de typage des processus récursifs est quant à elle de la forme :

$$\text{(T-REC)}\quad \frac{\Gamma \vdash_s (s, V) : \mathsf{R} \qquad \Gamma; \langle (Z, X) : \mathsf{R}\rangle @Z \mid \Delta; Z : \mathsf{R} \vdash_Z P}{\Gamma \mid \Delta \vdash_s (\mathsf{rec}\ Z : \mathsf{R}\,(X)\,.\,P)\,\langle V\rangle}$$

La continuation est typée dans l'environnement $\Gamma; \langle (Z, X) : \mathsf{R}\rangle @Z$ formé de la même façon que pour les réceptions dans la règle (T-R). Dans ce cas, l'expansion de $(Z, X) : \mathsf{R}$ pour obtenir un environnement est effectuée dans la « localité » Z, pour la raison que nous avons exposée précédemment.

Remarque 2.22. L'indication de la localité où l'expansion est effectuée ne sert que pour les types de canaux locaux C. Or il se trouve que le couple (Z, X) contient explicitement cette localité et que le type R est toujours de la forme $\sum x : \textsc{loc}.\ \mathsf{T}$. Les types de canaux locaux C apparaissant dans T peuvent donc être remplacés par leur variante localisée à $x : \mathsf{C}@x$. On utilisera au chapitre 3 le fait que le type R peut ne contenir aucun type de canal non localisé.

L'expansion avait déjà été utilisée pour la sémantique de $\mathsf{newloc}\, k, (\vec{c})$: $\mathsf{L}\ \mathsf{with}\ P_k\ \mathsf{in}\ P$, dans un cas particulier. En effet, on simplifiera la notation en $\langle X : \mathsf{T}\rangle$ dans le cas particulier où la localité courante ne joue aucun rôle dans cette expansion, comme c'est le cas pour les types de la forme L : étant donné que tous les types de canaux y sont explicitement localisés, le cas de la définition $\langle u : \mathsf{C}\rangle \,@t$ n'apparaît jamais à l'expansion. Cette notation simplifiée est par conséquent aussi utilisée dans la règle (T-NEWLOC).

La règle (T-IF) contient une dernière notation non expliquée : $[\Gamma]_{u_1=u_2}$. Voyons pourquoi cette notation apparaît dans le typage de P_1. Le processus $\mathsf{if}\ u_1 = u_2\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2$ sera bien formé dans un environnement Γ si les processus P_1 et P_2 sont eux-mêmes bien formés dans Γ. Mais la sémantique de cette primitive du langage, telle que décrite figure 1.4 impose que ce processus ne puisse se réduire en P_1 que si les identifiants u_1 et u_2 ont été égalisés à l'exécution. Dans le cas d'égalité, quel que soit le nom par lequel ils seront remplacés à

Fig. 2.11 Extension substitutive

$$
\begin{array}{rcl}
[\Gamma, s_i : \mathsf{E}]_{s_1=s_2} & = & [\Gamma]_{s_1=s_2}, s_1 : \mathsf{E}, s_2 : \mathsf{E} \\
[\Gamma, s : \mathsf{C}_{@} s_i]_{s_1=s_2} & = & [\Gamma]_{s_1=s_2}, s : \mathsf{C}_{@} s_1, s : \mathsf{C}_{@} s_2 \quad \text{si } s \notin \{s_1, s_2\} \\
[\Gamma, s : \mathsf{E}]_{s_1=s_2} & = & [\Gamma]_{s_1=s_2}, s : \mathsf{E} \quad \text{dans les autres cas}
\end{array}
$$

l'exécution, ce nom réunira au moins les capacités de u_1 et de u_2. Considérons un exemple très simple $l[\![\ldots \mathsf{if}\ u_1 = u_2\ \mathsf{then}\ a\,!\,\langle u_1\rangle\ \mathsf{else}\ \mathsf{stop}]\!]$ dans l'environnement $l : \textsc{loc}, a : \textsc{w}\langle \textsc{rw}\langle\rangle_{@} l, u_1 : \textsc{r}\langle\rangle_{@} l, u_2 : \textsc{w}\langle\rangle_{@} l$. Pour être communiqué sur le canal a un nom doit fournir les deux capacités d'émission et de réception. Or les deux identifiants u_1 et u_2 sont connus avec l'une des deux capacités seulement. On aimerait donc pouvoir prouver que quand les deux noms coïncident, l'émission de u_1 (ou de u_2, la situation étant symétrique) sur le canal a ne provoquera pas d'erreur. Pour cela, au lieu de vérifier le processus P_1 simplement dans l'environnement Γ, on le typera dans l'environnement $[\Gamma]_{u_1=u_2}$ dans lequel toutes les hypothèses mentionnant u_1 sont aussi retranscrites pour porter sur u_2, et réciproquement. Cette opération ne s'applique que sur l'environnement de typage Γ et non sur la liste Δ car les types R apparaissant dans Δ ne contiennent aucun identifiant libre. Cette simple transposition est définie formellement figure 2.11.

La définition de la figure 2.11 soulève cependant un problème. Imaginons le système

$$l[\![c\,?\left((x, y) : \sum x : \textsc{loc}.\ \textsc{w}\langle\rangle_{@} x\right) \mathsf{if}\ l = x\ \mathsf{then}\ y\,!\,\langle\rangle\ \mathsf{else}\ \mathsf{goto}\ x.\ y\,!\,\langle\rangle]\!]$$

qui attend un canal et sa localité ; si le canal est situé dans la localité actuelle, il est utilisé directement, sinon le processus migre avant de l'utiliser. Si on tente de typer ce système dans l'environnement $\Gamma = l : \textsc{loc}, c : \textsc{r}\langle \sum x : \textsc{loc}.\ \textsc{w}\langle\rangle_{@} x\rangle_{@} l$, on devra prouver

$$\Gamma; x : \textsc{loc}, y : \textsc{w}\langle\rangle_{@} x \mid \vdash_l \mathsf{if}\ l = x\ \mathsf{then}\ y\,!\,\langle\rangle\ \mathsf{else}\ \mathsf{goto}\ x.\ y\,!\,\langle\rangle$$

soit

$$[\Gamma; x : \textsc{loc}, y : \textsc{w}\langle\rangle_{@} x]_{l=x} \mid \vdash_l y\,!\,\langle\rangle$$

où $[\Gamma; x : \textsc{loc}, y : \textsc{w}\langle\rangle_{@} x]_{l=x}$ contient à la fois les hypothèses $y : \textsc{w}\langle\rangle_{@} l$ et $y : \textsc{w}\langle\rangle_{@} x$. Malheureusement, les types $\textsc{w}\langle\rangle_{@} l$ et $\textsc{w}\langle\rangle_{@} x$ ne sont pas compatibles car l et x ne sont pas syntaxiquement égaux. Les règles de formation des environnements données figure 2.7 interdisent donc cet environnement. On veut cependant pouvoir typer le processus $y\,!\,\langle\rangle$ en sachant qu'il est situé dans l : ces circonstances qui associent les deux types $\textsc{w}\langle\rangle_{@} l$ et $\textsc{w}\langle\rangle_{@} x$ à un même identifiant ne pourront concerner que des morceaux du processus où il sera certain que la variable x a effectivement été substituée par le nom l au cours de l'exécution. Pour autoriser de tels environnements au typage, on assouplira donc les contraintes de compatibilité dans la formation des environnements en parlant de *compatibilité large*.

Définition 2.23 (Compatibilité large)**.** *On dit que les types* T_1 *et* T_2 *sont* compatibles au sens large *si :*

– *ils sont compatibles ;*
– *ou ils sont tous les deux de la forme* $\mathsf{C}_i @ s_i$*, les types* C_i *sont compatibles et les* s_i *ne sont pas tous les deux des noms.*

On réutilisera la notation $\mathsf{T}_1 \downarrow \mathsf{T}_2$ pour la compatibilité large et on réinterprétera les règles de formation des environnements de la figure 2.7 avec cette notion large de compatibilité. On peut réinterpréter ainsi les règles de formation dans la mesure où cette modification de la notion de compatibilité ne change en rien le typage des systèmes : l'élargissement de la compatibilité n'intervient que lorsque des variables apparaissent dans les types. Puisque tous les systèmes seront supposés clos, des variables ne pourront être libres qu'au cours du typage des processus et les règles de typage prêtent particulièrement attention à n'utiliser la compatibilité large sur des types $\mathsf{C}@s$ que quand elles se placent dans le cas où les s se révéleront être le même nom à l'exécution. En effet, les deux seules règles qui peuvent introduire des hypothèses utilisant l'aspect « large » de la compatibilité sont (T-IF) et (T-HERE). Dans le cas (T-IF), l'égalité des noms à l'exécution est garantie par la règle de réduction du test. Pour (T-HERE) qui définit le typage des processus de la forme $\mathsf{here}\,(x)\,P$, la convention de Barendregt assure que la variable x est distincte de toutes les variables de l'environnement Γ, la notation $[\Gamma]_{s=x}$ permet donc de désigner simplement l'environnement dans lequel toutes les connaissances sur la localité courante s sont retranscrites pour x. Dans ce cas également, la règle de réduction (R-HERE) du préfixe $\mathsf{here}\,(x)$ garantira automatiquement que la localité courante et x coïncident à l'exécution.

Mais l'élargissement de la notion de compatibilité n'est bien entendu pas suffisant pour assurer que, partant d'un environnement bien formé Γ, $[\Gamma]_{s_1=s_2}$ le soit aussi. Par exemple $[l : \textsc{loc}, c : \textsc{r}\langle\rangle@l]_{l=c}$ sera l'environnement insensé $l : \textsc{loc}, c : \textsc{loc}, l : \textsc{r}\langle\rangle@l, c : \textsc{r}\langle\rangle@l$. Mais, comme l'intuition le veut, les environnements obtenus ne s'avéreront ainsi mal formés que dans des cas où les deux noms comparés ne peuvent pas être égaux, sans quoi leurs types ne seraient pas incohérents. On prouvera en effet par la suite (lemme 4.11) que l'environnement $[\Gamma]_{a=a}$, c'est-à-dire le véritable environnement utilisé à l'exécution quand il y a égalité, est équivalent à Γ.

Par ailleurs la règle de typage des tests que l'on obtient finalement n'impose nullement que les noms ou variables comparés apparaissent dans l'environnement de typage. Alors que toutes les hypothèses d'un environnement indiquent nécessairement une capacité de l'identifiant sur lequel elles portent, cet aspect assez surprenant du typage de la condition permet de mentionner un nom dans un processus sans en connaître plus que le nom lui-même. Cela s'avérera particulièrement significatif dans le cadre des équivalences typées[4].

Les autres règles de typage des processus, en particulier sur les émissions et réceptions, effectuent les vérifications introduites dans la présentation du typage. Plus précisément, pour qu'une émission se fasse sans risques, il suffit que la valeur émise ait le type des valeurs qui peuvent y être émises ; et le type d'un canal indique aussi le type auquel les valeurs sont reçues, c'est-à-dire de quelle façon l'environnement sera enrichi après cette réception. En outre, lorsque de nouveaux noms sont générés par les constructions newloc ou newchan, ils sont simplement rajoutés à l'environnement afin de pouvoir être utilisés dans le typage des continuations. Le typage de la construction goto implique

[4]Cela signifie en particulier que les noms libres apparaissant dans les actions sont distinguables, même pour un observateur ne disposant d'aucune capacité pour ces noms.

Fig. 2.12 Typage des systèmes

$$\text{(T-NIL)}\ \frac{\Gamma \vdash \mathbf{env}}{\Gamma \vdash \mathbf{0}} \qquad \text{(T-NEW)}\ \frac{\Gamma; a : \mathsf{E} \vdash M}{\Gamma \vdash (\mathsf{new}\, a : \mathsf{E})\, M}$$

$$\text{(T-PROC)}\ \frac{\Gamma \vdash l : \text{LOC} \quad \Gamma \mid \vdash_l P}{\Gamma \vdash l[\![P]\!]} \qquad \text{(T-S-PAR)}\ \frac{\Gamma \vdash M_1 \quad \Gamma \vdash M_2}{\Gamma \vdash M_1 \mid M_2}$$

uniquement la vérification que la destination de cette migration est une localité. Les règles (T-RÉP) et (T-PAR) quant à elles se contentent de reporter la question de bonne formation sur les continuations. Enfin (T-STOP) se charge d'assurer que l'environnement dans lequel les vérifications se passent ne contient aucune incohérence. Comme tous les processus s'achèvent par stop, cette règle permettra de prouver qu'un processus n'est bien typé que dans un environnement lui-même bien formé, ce qui évitera d'avoir des processus faussement bien typés à cause d'un environnement mal formé.

Comme prévu, le typage des processus permet de déduire un typage des systèmes. La principale règle pour les systèmes affirme que le système $l[\![P]\!]$ est bien typé dans l'environnement Γ dès lors que l est effectivement une localité dans Γ et que P est bien typé dans l pour l'environnement Γ. Puisque les variables de récursion doivent être liées dans les processus, l'ensemble Δ de ces variables sera considéré comme vide. Le prérequis au jugement $\Gamma \vdash l[\![P]\!]$ prend donc la forme $\Gamma \mid \vdash_l P$. Les autres règles, données figure 2.12, sont naturelles. Comme l'illustre principalement (T-NIL), qui assure le typage du système $\mathbf{0}$, ces règles imposent que l'environnement Γ soit bien formé pour que le jugement soit possible.

2.8 Propriétés du typage

Avant de vérifier que le typage accomplit bien les missions pour lesquelles il a été introduit, nous allons établir quelques propriétés fondamentales du typage. En particulier, un jugement de typage ne peut être prouvé que dans un environnement bien formé.

Proposition 2.24 (Le typage suppose un environnement bien formé)**.** *Soit un environnement* Γ *tel que l'un des jugements suivants soit prouvable :*

- $\Gamma; \Gamma' \vdash \mathbf{env}$ *pour un* Γ' *quelconque ;*
- $\Gamma \vdash S : \mathsf{T}$ *pour des* S *et* T *quelconques ;*
- $\Gamma \vdash_s S : \mathsf{T}$ *pour des* s*,* S *et* T *quelconques ;*
- $\Gamma \mid \Delta \vdash_s P$ *pour des* Δ*,* s *et* P *quelconques ;*
- $\Gamma \vdash M$ *pour un* M *quelconque.*

Alors $\Gamma \vdash \mathbf{env}$*.*

Démonstration. Cette preuve repose sur une simple induction sur la preuve de $\Gamma \vdash M$, $\Gamma \mid \Delta \vdash_s P$, $\Gamma \vdash_s V : \mathsf{T}$, $\Gamma \vdash s : \mathsf{E}$ ou $\Gamma; \Gamma' \vdash \mathbf{env}$ car tous les cas de base ((T-NIL), (T-STOP), etc.) supposent la bonne formation de leur environnement. □

Par ailleurs, le sous-typage défini sur les types peut être étendu assez naturellement aux environnements entiers. Comme sur les types, le sous-typage sur les environnements engendre au passage une notion d'équivalence.

Définition 2.25 (Sous-typage et équivalence d'environnements)**.** *Soient deux environnements bien formés Γ_1 et Γ_2. On dit que Γ_1 est un environnement sous-type de Γ_2, noté $\Gamma_1 <: \Gamma_2$, si pour toute hypothèse $s : \mathsf{T}$ de Γ_2, $\Gamma_1 \vdash s : \mathsf{T}$ est prouvable.*

Si Γ_1 n'est pas bien formé mais qu'il existe $s_1 : \text{LOC}, \ldots, s_n : \text{LOC}$ tel que $s_1 : \text{LOC}, \ldots, s_n : \text{LOC}; \Gamma_1 \vdash \mathbf{env}$, on dit que Γ_1 est un sous-type de Γ_2 si $s_1 : \text{LOC}, \ldots, s_n : \text{LOC}; \Gamma_1 <: s_1 : \text{LOC}, \ldots, s_n : \text{LOC}; \Gamma_2$.

On dit que Γ_1 et Γ_2 sont équivalents, noté $\Gamma_1 \equiv \Gamma_2$, si $\Gamma_1 <: \Gamma_2$ et $\Gamma_2 <: \Gamma_1$.

Au même titre qu'une valeur d'un sous-type de T peut être utilisée lorsqu'une valeur de type T est attendue, un environnement sous-type permet de prouver tous les jugements vrais dans un environnement sur-type. Intuitivement, le système $l[\![c\,!\,\langle\rangle]\!]$ sera en effet bien typé dès que le canal c est disponible en écriture, donc en particulier aussi dans les environnements qui associent à c le type $\text{RW}\langle\rangle@l$.

Proposition 2.26 (Affaiblissement)**.** *Soient Γ_1 et Γ_2 deux environnements bien formés tels que $\Gamma_1 <: \Gamma_2$. Alors :*

1. *$\Gamma_2 \vdash s : \mathsf{E}$ implique $\Gamma_1 \vdash s : \mathsf{E}$;*
2. *$\Gamma_2 \vdash_s V : \mathsf{T}$ implique $\Gamma_1 \vdash_s V : \mathsf{T}$;*
3. *$\Gamma_2 \mid \Delta \vdash_s P$ implique $\Gamma_1 \mid \Delta; \Delta' \vdash_s P$;*
4. *$\Gamma_2 \vdash M$ implique $\Gamma_1 \vdash M$.*

Cette proposition sera démontrée en détails dans le cadre plus complexe du chapitre 4. Elle implique que tous les jugements de typage prouvables dans un environnement le sont aussi dans tout environnement équivalent.

Corollaire 2.27. *Soient Γ_1 et Γ_2 deux environnements bien formés tels que $\Gamma_1 \equiv \Gamma_2$. Alors :*

1. *$\Gamma_1 \vdash s : \mathsf{E}$ si et seulement si $\Gamma_2 \vdash s : \mathsf{E}$;*
2. *$\Gamma_1 \vdash_s V : \mathsf{T}$ si et seulement si $\Gamma_2 \vdash_s V : \mathsf{T}$;*
3. *$\Gamma_1 \mid \Delta \vdash_s P$ si et seulement si $\Gamma_2 \mid \Delta; \Delta' \vdash_s P$;*
4. *$\Gamma_1 \vdash M$ si et seulement si $\Gamma_2 \vdash M$.*

Cette proposition permet aussi de prouver que la congruence structurelle ne met en relation que des processus qui sont typables sous les mêmes hypothèses. Cette cohérence, dont la preuve sera, elle aussi, fournie plus tard, peut se formaliser ainsi :

Proposition 2.28 (Cohérence du typage avec $\equiv$)**.** *Soient deux systèmes M et N tels que $M \equiv N$. Quel que soit l'environnement Γ, $\Gamma \vdash M$ si et seulement si $\Gamma \vdash N$.*

Ces propriétés du système de types permettent d'étudier le contrôle que le typage apporte au calcul. Une des questions qui se posent est de savoir si un système bien typé peut arriver dans un état incohérent, au sens intuitif des dangers esquissés au début de ce chapitre. Trois principales sources d'erreurs peuvent être identifiées :

1. un processus qui utiliserait un nom à la fois comme localité et comme canal, par exemple le processus $a[\![a\,!\,\langle\rangle]\!]$;
2. deux processus d'un système qui tenteraient de communiquer mais sans respecter la même structure de message, c'est-à-dire l'un émettant un message V sur un canal sur lequel l'autre attend un motif X dans un cas où la décomposition arborescente simultanée de (V, X) n'est pas définie ;
3. une incohérence similaire entre l'argument et le paramètre d'un processus récursif.

On voit assez facilement comment ces erreurs seront impossibles dans des processus bien typés : $a[\![a\,!\,\langle\rangle]\!]$ supposera par exemple à la fois $a : \text{LOC}$ et $a : \text{W}\langle\rangle_{@a}$, deux hypothèses totalement incompatibles donc ne pouvant coexister dans un environnement de typage. Les deux autres sortes d'erreurs sont empêchées par le fait qu'un même type sera associé à V et X et que ce type commun entraîne naturellement l'existence de la décomposition arborescente simultanée.

Le typage a également pour vocation de contrôler les capacités, c'est-à-dire que les processus bien typés n'utilisent jamais des capacités dont ils ne disposent pas. Toutes ces garanties, assurant qu'un processus bien typé n'effectue pas de réductions erronées ou illicites, se formalisent en développant une variante du calcul où les processus sont étiquetés par un environnement de typage et où deux sortes de réductions sont définies : les normales, correspondant à la sémantique par réductions du calcul, et les prohibées, englobant tous les autres cas de figure. Cette formalisation nous entraînerait loin des points essentiels auxquels on s'intéresse dans ce travail, mais elle est développée plus précisément dans le cadre du π-calcul dans [PS96] et dans le cadre du Dπ-calcul avec typage des capacités dans [HR02], deux systèmes de types dont découle le typage présenté ici.

La sûreté qu'assure ainsi le typage ne porte que sur les réductions immédiatement possibles pour un système bien typé. Mais le typage a vocation à autoriser une analyse statique des systèmes qui garantisse malgré tout des propriétés valables tout au long de leur exécution. Il est par conséquent central de pouvoir assurer que cette sûreté est conservée au cours du calcul.

Théorème 2.29 (Préservation du typage)**.** *Soit un système M bien typé dans l'environnement Γ et tel qu'il existe M' avec $M \longrightarrow M'$. Alors M' est bien typé dans Γ.*

La démonstration de ce théorème nécessite la mise en place de lemmes pour les substitutions de variables et variables de récursion qui sont mises en jeu lors des communications et des déroulements des appels récursifs. Encore une fois, ces preuves (y compris le découpage en lemmes) sont des cas simplifiés de celles fournies au chapitre 4.

Regardons maintenant l'impact du typage sur l'équivalence observationnelle définie au chapitre 1.

2.9 Équivalence observationnelle typée

La distinction entre les capacités d'émission et de réception de messages sur les canaux, avec le sous-typage qu'elle entraîne, modifie fortement la notion intuitive d'équivalence observationnelle pour les systèmes. En effet, l'idée maîtresse de

la notion d'équivalence définie à la section 1.3 est de distinguer des systèmes uniquement quand un observateur peut faire cette distinction en interagissant avec ces systèmes. Regardons donc comment les idées maîtresses de l'équivalence observationnelle (définition 1.12) peuvent être adaptées à un cadre typé.

Deux conditions doivent être remplies pour qu'une interaction soit possible entre des systèmes typés : il faut connaître à la fois le canal sur lequel cette interaction prend place mais aussi avoir la capacité nécessaire pour interagir. Considérons par exemple le système

$$l[\![\mathsf{newchan}\, c_1 : \mathrm{RW}\langle \mathrm{W}\langle\rangle\rangle \,\mathsf{in}\, \mathsf{newchan}\, c_2 : \mathrm{RW}\langle\rangle \,\mathsf{in}\, c\,!\,\langle c_1\rangle\, c_1\,!\,\langle c_2\rangle\, c_2\,!\,\langle\rangle]\!]$$

En supposant que l'observateur connaisse la localité l et le canal c, il pourra donc interagir avec le système sur ce canal et ainsi apprendre le nom c_1, en plaçant ce système dans un contexte $l[\![c\,?\,(x : \mathsf{T})\, P]\!]\,|[\cdot]$. Même si l'observateur apprenait au passage toutes les capacités existant pour le nom c_1, c'est-à-dire au cas où le type T serait $\mathrm{RW}\langle \mathrm{W}\langle\rangle\rangle$, le seul moyen de poursuivre l'interaction avec le système serait de prendre un P de la forme $x\,?\,(x' : \mathrm{W}\langle\rangle)\, P'$. Dès lors, incapable de recevoir un message sur le canal x' qui serait remplacé par c_2 après la communication sur c_1, l'observateur ne pourrait pas interagir avec la continuation $c_2\,!\,\langle\rangle$. Par conséquent, quoi que fasse l'observateur, le système précédent exhibera exactement le même comportement que

$$l[\![\mathsf{newchan}\, c_1 : \mathrm{RW}\langle \mathrm{W}\langle\rangle\rangle \,\mathsf{in}\, \mathsf{newchan}\, c_2 : \mathrm{RW}\langle\rangle \,\mathsf{in}\, c\,!\,\langle c_1\rangle\, c_1\,!\,\langle c_2\rangle]\!]$$

la dernière émission, sur le canal c_2, étant invisible.

Essayons d'obtenir une équivalence observationnelle typée qui accorde les mêmes droits d'observations que l'équivalence de la section 1.3 dans la mesure où ces droits n'entrent pas en conflit avec les capacités décrites par le typage. Pour définir une équivalence qui respecte les limitations mises en lumière dans l'exemple précédent, il est nécessaire de restreindre les contextes dans lesquels l'observateur peut placer le système qu'il observe : il est indispensable qu'une fois le système observé placé dans le contexte de l'observateur, le tout soit encore bien typé. Par ailleurs, pour caractériser les messages émis par le système que l'observateur peut voir, il faudra que l'observateur dispose de la capacité de recevoir ce message. Partant de cette idée, une façon très simple et très pratique de décrire les connaissances de l'observateur est d'utiliser un environnement de typage que l'on notera Ω. Ces connaissances permettront aussi de caractériser précisément les contextes que l'observateur peut utiliser pour étudier un système. L'équivalence que nous allons définir prendra donc en compte à la fois les systèmes observés mais aussi la connaissance de leur observateur. Il faudra cependant que cette connaissance ne soit pas incohérente avec l'observé. En effet, si un observateur considère le système $l[\![c\,!\,\langle\rangle]\!]$ en supposant que c est un nom de localité, il pourra naturellement essayer d'utiliser un contexte $[\cdot]\,|\,c[\![P]\!]$, ce qui aboutirait à un système mal formé. Cette adéquation peut se formaliser par la simple contrainte suivante concernant l'environnement Ω et le système observé :

Définition 2.30 (Configuration)**.** *On appelle* configuration *toute paire formée d'un environnement Ω et d'un système M, notée $\Omega \rhd M$, telle qu'il existe un environnement Γ, sous-type de Ω, dans lequel M est bien typé.*

De la même façon, les relations d'équivalence considérées dans la suite mentionneront l'environnement représentant la connaissance de l'observateur et

imposeront que cette connaissance soit cohérente avec les systèmes mis en relation.

Définition 2.31 (Relation typée)**.** *On appellera* relation typée *une relation $\mathcal{R}$ portant sur des triplets (Ω, M, N) telle que $\Omega \rhd M$ et $\Omega \rhd N$ soient des configurations.*

On notera $\Omega \vDash M\ \mathcal{R}\ N$ l'appartenance du triplet (Ω, M, N) à la relation $\mathcal{R}$.

Ces notions génériques permettent de formaliser la notion d'équivalence typée que nous cherchons. Commençons donc par définir une notion typée de barbe.

Définition 2.32 (Barbe typée)**.** *On dit que la configuration $\Omega \rhd M$ exhibe une* barbe typée *sur a, ce que l'on note $\Omega \rhd M \Downarrow a$, si et seulement s'il existe une localité l telle que :*

- *$\Omega \vdash a : \mathrm{R}\langle \mathsf{T} \rangle_@ l$, pour un type T quelconque,*
- *et il existe V, P, M' et Φ tels que*

$$M \Longrightarrow\equiv (\mathit{new}\,\Phi)(M' \mid l[\![a\,!\,\langle V\rangle\ P]\!])$$

avec $a, l \notin \mathsf{dom}(\Phi)$.

Cette définition montre une différence notable avec celle des barbes non typées : la localité dans laquelle la barbe est exhibée n'est pas mentionnée explicitement dans $\Omega \rhd M \Downarrow a$. La raison en est que l'environnement Ω doit associer à a un type de la forme $\mathrm{R}\langle \mathsf{T} \rangle_@ l$ qui impose donc la localité l où cette barbe apparaît.

Le respect des barbes typées est défini à partir des barbes typées exactement comme le respect des barbes non typées.

Définition 2.33 (Respect des barbes typées[5])**.** *On dit qu'une relation typée $\mathcal{R}$ respecte les barbes typées quand $\Omega \vDash M\ \mathcal{R}\ N$ implique que, quel que soit a, $\Omega \rhd M \Downarrow a$ si et seulement si $\Omega \rhd N \Downarrow a$.*

Par contre, la notion de contextualité pour une relation typée est moins directe à formaliser. L'environnement Ω permet de caractériser simplement les contextes de la forme $M\,|[\cdot]$ dans lequel l'observateur peut placer le système observé : il suffit dans ce cas-là de vérifier que M soit bien typé dans Ω. Mais la question reste entière pour un contexte quelconque tel que :

$$(\mathsf{new}\,\vec{a}_1 : \vec{\mathsf{E}}_1)\,M_1\,|((\mathsf{new}\,\vec{a}_2 : \vec{\mathsf{E}}_2)\,M_2\,|\ldots|((\mathsf{new}\,\vec{a}_n : \vec{\mathsf{E}}_n)\,M_n\,|[\cdot])\ldots)$$

D'ailleurs, la difficulté apparaît déjà sur des contextes très simples. Ainsi, le processus P du contexte $(\mathsf{new}\,a : \mathsf{E})(l[\![P]\!]\,|[\cdot])$ est capable d'interagir sur le nom a avec le système placé dans ce contexte. Par conséquent le fait de savoir $\Omega \vDash M\ \mathcal{R}\ N$ ne permet pas de conclure que $\Omega \vDash (\mathsf{new}\,a : \mathsf{E})(l[\![P]\!] \mid M)\ \mathcal{R}\ (\mathsf{new}\,a : \mathsf{E})(l[\![P]\!] \mid N)$ dans le cas général. Pour prendre un exemple, il serait assez naturel de considérer qu'une relation d'équivalence typée $\mathcal{R}$ puisse contenir le triplet

$$l : \mathrm{LOC}, b : \mathrm{RW}\langle\rangle_@ l \vDash l[\![a\,!\,\langle\rangle]\!]\ \mathcal{R}\ l[\![\mathsf{stop}]\!]$$

même si elle ne pourra pas contenir le triplet

$$l : \mathrm{LOC}, b : \mathrm{RW}\langle\rangle_@ l \vDash (\mathsf{new}\,a : \mathrm{RW}\langle\rangle_@ l)\,l[\![a\,?\,()\,b\,!\,\langle\rangle]\!] \mid l[\![a\,!\,\langle\rangle]\!]\ \mathcal{R}\ (\mathsf{new}\,a : \mathrm{RW}\langle\rangle_@ l)\,l[\![a\,?\,()\,b\,!\,\langle\rangle]\!] \mid l[\![\mathsf{stop}]\!]$$

[5] Et des barbus typés ?

si elle respecte les barbes typées car $(\mathsf{new}\, a : \text{RW}\langle\rangle_{@} l)\, l[\![a\,?\,()\, b\,!\,\langle\rangle]\!] \mid l[\![a\,!\,\langle\rangle]\!]$ peut exhiber une barbe sur le canal b tandis que $(\mathsf{new}\, a : \text{RW}\langle\rangle_{@} l)\, l[\![a\,?\,()\, b\,!\,\langle\rangle]\!] \mid l[\![\mathsf{stop}]\!]$ en est incapable.

Par contre, si l'on sait que les deux systèmes M et N sont équivalents pour un observateur $\Omega, a : \mathsf{E}$, et que les configurations $\Omega \rhd (\mathsf{new}\, a : \mathsf{E})\, M$ et $\Omega \rhd (\mathsf{new}\, a : \mathsf{E})\, N$ sont bien formées, on peut conclure que les deux systèmes $(\mathsf{new}\, a : \mathsf{E})\, M$ et $(\mathsf{new}\, a : \mathsf{E})\, N$ sont équivalents pour l'observateur Ω.

Ce constat indique une marche à suivre : définir la contextualité des relations typées en décomposant les contextes. Les cas des contextes $(\mathsf{new}\, a : \mathsf{E})[\cdot]$ et $[\cdot] \mid l[\![P]\!]$ ont déjà été évoqués. Ces deux sortes de contextes émettent des suppositions sur l'environnement : pour le contexte $(\mathsf{new}\, a : \mathsf{E})[\cdot]$, il faudra que les deux systèmes observés M et N soient équivalents dans un environnement contenant l'hypothèse $a : \mathsf{E}$; pour $[\cdot] \mid l[\![P]\!]$, l'environnement devra être capable de typer $l[\![P]\!]$. Cependant, pour poursuivre sur l'exemple précédent, on voudrait pouvoir conclure de

$$l : \text{LOC}, b : \text{RW}\langle\rangle_{@} l \vDash l[\![a\,!\,\langle\rangle]\!] \;\mathcal{R}\; l[\![\mathsf{stop}]\!]$$

que

$$\begin{array}{l} l : \text{LOC}, b : \text{RW}\langle\rangle_{@} l \vDash \\ \quad (\mathsf{new}\, c : \text{R}\langle\rangle_{@} l)\, l[\![a\,!\,\langle\rangle]\!] \mid l[\![c\,!\,\langle\rangle]\!] \;\mathcal{R}\; (\mathsf{new}\, c : \text{R}\langle\rangle_{@} l)\, l[\![\mathsf{stop}]\!] \mid l[\![c\,!\,\langle\rangle]\!] \end{array}$$

La contextualité imposera pour cela la possibilité pour l'observateur de créer de nouveaux noms, c'est-à-dire d'enrichir ses connaissances dans la mesure où elles ne portent que sur des noms inédits donc n'apparaissant pas dans les observés.

En combinant ces différents points, la contextualité pourra donc se formaliser ainsi :

Définition 2.34 (Relation typée contextuelle)**.** *Une relation typée $\mathcal{R}$ entre systèmes est dite* contextuelle *si et seulement si les conditions suivantes sont vérifiées :*

- *Si $\Omega \vDash M\ \mathcal{R}\ N$, et Ω' est un environnement tel que $\mathsf{dom}(\Omega')$ ne contienne que des noms inédits alors $\Omega; \Omega' \vDash M\ \mathcal{R}\ N$.*
- *Si $\Omega \vDash M\ \mathcal{R}\ N$ et $\Omega \vdash k[\![P]\!]$ alors $\Omega \vDash M \mid k[\![P]\!]\ \mathcal{R}\ N \mid k[\![P]\!]$.*
- *Si $\Omega; a : \mathsf{E} \vDash M\ \mathcal{R}\ N$ et les configurations $\Omega \rhd (\mathsf{new}\, a : \mathsf{E})\, M$ et $\Omega \rhd (\mathsf{new}\, a : \mathsf{E})\, N$ sont bien formées, alors $\Omega \vDash (\mathsf{new}\, a : \mathsf{E})\, M\ \mathcal{R}\ (\mathsf{new}\, a : \mathsf{E})\, N$.*

Grâce à la notion de contextualité on peut définir une variante typée de la congruence barbue fermée par réduction.

Définition 2.35 (Congruence typée barbue fermée par réduction)**.** *On appelle* congruence typée barbue fermée par réduction *la plus grande relation typée symétrique contextuelle, fermée par réduction et qui respecte les barbes typées. On la note $\cong^{tbr}$.*

La preuve que cette relation est effectivement une équivalence fonctionne de façon très similaire au cas non typé. Cette relation est cependant nettement plus contrainte que $\cong^{br}$. En particulier le contexte $[\cdot] \mid ((\mathsf{new}\, a : \mathsf{E})\, M)$ n'est pas autorisé. Ce choix permet d'alléger la complexité de la preuve que deux systèmes sont équivalents : en effet, comme nous l'avions illustré pour $\cong^{br}$, une telle preuve doit prendre en compte tous les contextes possibles. Mais ce choix est

d'autant plus raisonnable qu'il ne modifie pas l'équivalence obtenue. En effet, si on considère la congruence structurelle typée, c'est-à-dire la relation $\equiv_t$ telle que $\Omega \vDash M \equiv_t N$ dès que $M \equiv N$ et que les configurations $\Omega \rhd M$ et $\Omega \rhd N$ sont bien formées, on montre aisément que cette relation est incluse dans $\cong^{tbr}$.

Proposition 2.36. $\equiv_t \subseteq \cong^{tbr}$

Démonstration. $\equiv_t$ est naturellement symétrique et fermée par réduction, par le théorème 2.29. De plus une analyse des règles qui définissent la congruence structurelle permet de prouver qu'elle respecte les barbes typées. Quant à la contextualité, la proposition 2.26 sur l'affaiblissement permet de garantir que $\Omega \vDash M \equiv_t N$ entraîne $\Omega;\Omega' \vDash M \equiv_t N$ dès lors que $\mathsf{dom}(\Omega') \cap (\mathsf{dom}(\Omega) \cup \mathsf{fn}(M) \cup \mathsf{fn}(N)) = \emptyset$; et la contextualité (non typée) de $\equiv$ donne les deux autres propriétés requises. □

La congruence structurelle typée permet donc de traiter simplement tous les contextes de la forme $[\cdot] \mid M_O$ puisque M_O est toujours structurellement congru à un système $(\mathsf{new}\,\Phi)\, l_1[\![P_1]\!] \mid \cdots \mid l_n[\![P_n]\!]$. Donc

$$\Omega \vDash M \mid M_O \equiv_t (\mathsf{new}\,\Phi)(M \mid l_1[\![P_1]\!] \mid \cdots) \cong^{tbr} (\mathsf{new}\,\Phi)(N \mid l_1[\![P_1]\!] \mid \cdots) \equiv_t N \mid M_O$$

permet de ramener le contexte aux cas apparaissant dans la définition de la contextualité typée.

Le choix de restreindre les observations rudimentaires aux messages émis par les systèmes observés (et non reçus) ne bride pas l'équivalence pour une raison similaire. En effet, si un système est capable de recevoir un message sur le canal a dans la localité l, le petit contexte $[\cdot] \mid l[\![a\,!\,\langle V\rangle\, b\,!\,\langle\rangle]\!]$, dans lequel le canal b est inédit et rajouté par l'observateur, transforme la réception sur le canal a en une barbe sur le canal b.

2.10 Bisimilarité typée

Exactement comme pour la congruence $\cong^{br}$, il est fastidieux de prouver que deux systèmes sont équivalents pour la congruence typée barbue fermée par réduction. Une telle preuve requiert en effet de considérer tous les contextes dans lesquels ces systèmes pourraient être plongés afin de vérifier qu'ils y exhibent le même comportement. Cependant, pour différencier les deux systèmes $l[\![c\,!\,\langle c_1\rangle]\!]$ et $l[\![c\,!\,\langle c_2\rangle]\!]$ un contexte de la forme $[\cdot] \mid k[\![P]\!]$ ou $[\cdot] \mid l[\![d\,?\,()]\!]$ ne sera pas pertinent. Par contre le contexte $[\cdot] \mid l[\![c\,?\,(x : \mathsf{T})\,\mathsf{if}\ x = c_1\ \mathsf{then}\ d\,!\,\langle\rangle\ \mathsf{else}\ \mathsf{stop}]\!]$ permet de caractériser précisément le comportement du système $l[\![c\,!\,\langle c_1\rangle]\!]$ en n'exhibant une barbe sur le canal d que dans le cas où le système observé envoie le message c_1 sur le canal c.

Cet exemple suggère de chercher à identifier le plus simple contexte possible qui caractérise le comportement d'un système donné. Or la seule forme d'interaction possible entre un système et son contexte est une communication, toutes les autres réductions n'impliquant qu'un unique processus. Il y a donc deux formes de contextes caractéristiques : celle qui identifie une émission, comme dans l'exemple précédent, et celle qui identifie une réception.

Cette analyse permet de donner une définition alternative de la sémantique du calcul. Au lieu d'indiquer les réductions qu'un système est capable de faire, on

peut dire quelles seraient les réductions possibles, à la condition que le contexte dans lequel ce système est placé le permette. Poursuivant l'idée avancée dans le cas de la congruence typée barbue et fermée par réduction, ces aptitudes d'interactions du contexte seront caractérisées par un environnement de typage Ω. Par conséquent cette reformulation de la sémantique du langage parlera des transitions que les configurations peuvent effectuer. On notera ainsi

$$\Omega \rhd M \xrightarrow{\mu} \Omega' \rhd M'$$

si un observateur connaissant Ω peut, en plaçant le système M dans un contexte représenté par l'*action* μ, autoriser une interaction au cours de laquelle le système M évolue en M' et la connaissance de l'observateur est éventuellement enrichie pour devenir Ω'. Pour prendre un exemple très simple, on autorisera la transition

$$l : \text{LOC}, c : \text{R}\langle\rangle_@ l \rhd l[\![c\,!\,\langle\rangle\, P]\!] \xrightarrow{c!} l : \text{LOC}, c : \text{R}\langle\rangle_@ l \rhd l[\![P]\!]$$

qui signifie qu'un observateur connaissant la localité l et pouvant recevoir des messages sur le canal c de cette localité pourra placer le système $l[\![c\,!\,\langle\rangle\, P]\!]$ dans le plus simple contexte permettant de détecter l'émission sur le canal c. Dans cette transition, il est fondamental que l'observateur dispose de la capacité $\text{R}\langle\rangle_@ l$ pour le canal c pour réellement interagir avec le système quand le « signal » est émis. Par ailleurs, l'étiquette rajoutée sur la transition est ici $c!$. Elle mime donc directement le préfixe agissant, plutôt qu'elle ne décrit *in extenso* le contexte que l'observateur pourrait utiliser pour interagir avec cette émission.

Cette sémantique peut alors être définie par un jeu de règles sur la syntaxe des systèmes, appelé *système typé de transitions étiquetées* (STTE). Ces règles sont données à la figure 2.13.

Puisque le STTE est créé dans le but d'obtenir une nouvelle formulation de la congruence barbue typée, il est nécessaire de vérifier que la sémantique engendrée par le STTE coïncide avec la sémantique par réductions. Cette propriété sera prouvée dans le cadre du chapitre 4 mais on compare déjà ici les deux sémantiques.

Dans les cas où le système est capable de se réduire de lui-même, par exemple $l[\![\mathsf{goto}\, k.\ P]\!]$ qui devient $k[\![P]\!]$, le contexte minimal dans lequel un observateur doit le placer pour activer cette transition est bien évidemment $[\cdot]$. Dans ce cas-là, la transition est simplement étiquetée par un τ, afin de symboliser le fait qu'elle s'effectue de façon *invisible*, c'est-à-dire sans interaction. Pour ces constructions, le STTE est donc très proche de la sémantique par réductions.

En revanche, la sémantique du STTE est compositionnelle, contrairement à la sémantique par réductions. En effet, pour autoriser une communication sur un canal, la sémantique par réductions exige que les deux processus soient syntaxiquement « voisins », en reposant donc sur la congruence structurelle pour réaménager le système suivant les besoins. Par exemple, dans le système

$$l[\![a\,?\,(x : \mathsf{T})\, P_1]\!] \,|\, (\mathsf{new}\, b : \mathsf{E}_b)\, l[\![a\,!\,\langle b\rangle\, P_2]\!]$$

la communication sur le canal a permet au processus P_1 d'apprendre le nom b alors que la portée de celui-ci ne couvre initialement pas P_1. La sémantique par réductions traite la question en constatant que ce système est structurellement congru à

$$(\mathsf{new}\, b : \mathsf{E}_b)\, l[\![a\,?\,(x : \mathsf{T})\, P_1]\!] \,|\, l[\![a\,!\,\langle b\rangle\, P_2]\!]$$

dans lequel la question ne se pose plus.

Fig. 2.13 Système typé de transitions étiquetées

$$\text{(TE-W)}\quad \frac{\Omega \vdash_l a : \mathrm{R}\langle \mathsf{T}\rangle \quad \text{avec } \mathsf{T} = \Omega^r(a)}{\Omega \rhd l[\![a\,!\,\langle V\rangle\, P]\!] \xrightarrow{a!V} \Omega, \langle V : \mathsf{T}\rangle_{@l} \rhd l[\![P]\!]}$$

$$\text{(TE-R)}\quad \frac{\begin{array}{c}\Omega \vdash_l a : \mathrm{W}\langle \mathsf{T}'\rangle \\ \Omega \vdash_l V : \mathsf{T}'\end{array}}{\Omega \rhd l[\![a\,?\,(X : \mathsf{T})\, P]\!] \xrightarrow{a?V} \Omega \rhd l[\![P\{^V/_X\}]\!]}$$

$$\text{(TE-COMM)}\quad \frac{\begin{array}{c}\Omega_M \rhd M \xrightarrow{(\Phi)a!V} \Omega'_M \rhd M' \\ \Omega_N \rhd N \xrightarrow{(\Phi)a?V} \Omega'_N \rhd N'\end{array}}{\begin{array}{l}\Omega \rhd M \mid N \xrightarrow{\tau} \Omega \rhd (\mathsf{new}\,\Phi)\, M' \mid N' \\ \Omega \rhd N \mid M \xrightarrow{\tau} \Omega \rhd (\mathsf{new}\,\Phi)\, N' \mid M'\end{array}}$$

$$\text{(TE-GOTO)}\quad \Omega \rhd l[\![\mathsf{goto}\, k.\ P]\!] \xrightarrow{\tau} \Omega \rhd k[\![P]\!]$$

$$\text{(TE-IF-V)}\quad \Omega \rhd l[\![\mathsf{if}\ a = a\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2]\!] \xrightarrow{\tau} \Omega \rhd l[\![P_1]\!]$$

$$\text{(TE-IF-F)}\quad \Omega \rhd l[\![\mathsf{if}\ a_1 = a_2\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2]\!] \xrightarrow{\tau} \Omega \rhd l[\![P_2]\!] \quad \text{si } a_1 \neq a_2$$

$$\text{(TE-NEWCHAN)}\quad \Omega \rhd l[\![\mathsf{newchan}\, c : \mathsf{C}\ \mathsf{in}\ P]\!] \xrightarrow{\tau} \Omega \rhd (\mathsf{new}\, c : \mathsf{C}_{@l})\, l[\![P]\!]$$

$$\text{(TE-NEWLOC)}\quad \begin{array}{l}\Omega \rhd l[\![\mathsf{newloc}\, k, (\vec{c}) : \mathsf{L}\ \mathsf{with}\ P_k\ \mathsf{in}\ P]\!] \\ \qquad \xrightarrow{\tau} \Omega \rhd (\mathsf{new}\langle (k, (\vec{c})) : \mathsf{L}\rangle)\, k[\![P_k]\!] \mid l[\![P]\!]\end{array}$$

$$\text{(TE-SPLIT)}\quad \Omega \rhd l[\![P_1 \mid P_2]\!] \xrightarrow{\tau} \Omega \rhd l[\![P_1]\!] \mid l[\![P_2]\!]$$

$$\text{(TE-HERE)}\quad \Omega \rhd l[\![\mathsf{here}\,(x)\, P]\!] \xrightarrow{\tau} \Omega \rhd l[\![P\{^l/_x\}]\!]$$

$$\text{(TE-REP)}\quad \Omega \rhd l[\![*P]\!] \xrightarrow{\tau} \Omega \rhd l[\![P]\!] \mid l[\![*P]\!]$$

$$\text{(TE-REC)}\quad \Omega \rhd l[\![(\mathsf{rec}\ Z : \mathsf{R}\,(X)\,.\ P)\,\langle V\rangle]\!] \xrightarrow{\tau} \Omega \rhd l[\![P\{^V/_X\}[^{\mathsf{rec}\ Z:\mathsf{R}(X).\ P}/_Z]]\!]$$

$$\text{(TE-C-PAR)}\quad \frac{\Omega \rhd M \xrightarrow{\mu} \Omega' \rhd M'}{\begin{array}{l}\Omega \rhd M \mid N \xrightarrow{\mu} \Omega' \rhd M' \mid N \\ \Omega \rhd N \mid M \xrightarrow{\mu} \Omega' \rhd N \mid M'\end{array}}$$

$$\text{(TE-C-NEW)}\quad \frac{\Omega \rhd M \xrightarrow{\mu} \Omega' \rhd M'}{\Omega \rhd (\mathsf{new}\, a : \mathsf{E})\, M \xrightarrow{\mu} \Omega' \rhd (\mathsf{new}\, a : \mathsf{E})\, M'} \quad a \notin \mathsf{n}(\mu)$$

$$\text{(TE-OUV)}\quad \frac{\Omega \rhd M \xrightarrow{(\Phi)a!V} \Omega' \rhd M'}{\Omega \rhd (\mathsf{new}\, b : \mathsf{E})\, M \xrightarrow{(b:\mathsf{E};\Phi)a!V} \Omega' \rhd M'} \quad \begin{array}{l} b \neq a \\ b \in \mathsf{fn}(V)\end{array}$$

$$\text{(TE-AFF)}\quad \frac{\Omega; b : \mathsf{E} \rhd M \xrightarrow{(\Phi)a?V} \Omega' \rhd M'}{\Omega \rhd M \xrightarrow{(b:\mathsf{E};\Phi)a?V} \Omega' \rhd M'} \quad b \notin \{a\} \cup \mathsf{fn}(M)$$

Le système typé de transitions étiquetées permet, lui, de calculer la sémantique du système $M \mid N$ en composant simplement les sémantiques des systèmes M et N. Il apparaîtra ainsi qu'un processus situé dans le système M peut communiquer avec un autre situé dans N dès que le système M effectue une transition étiquetée[6] par $(\Phi)a!V$ et N par $(\Phi)a?V$, ou vice versa. En effet, la première étiquette symbolise l'aptitude d'un processus dans M à émettre sur le canal a le message V, émission qui nécessite l'extension des portées des différents noms apparaissant dans l'environnement Φ. De façon similaire, la seconde étiquette correspond à l'aptitude duale d'un processus de N, aptitude qui impose en particulier que les noms apparaissant dans le domaine de Φ ne soient pas libres dans N. Dès qu'il est possible de prouver que ces deux systèmes peuvent ainsi effectuer ces transitions duales, quels que soient les environnements Ω_M et Ω_N utilisés pour arriver à ces conclusions, on sait qu'ils pourront interagir.

L'environnement Φ apparaissant dans ces étiquettes pallie ainsi l'absence de congruence structurelle, utilisée constamment dans la sémantique par réductions. L'approche compositionnelle du STTE propose à la place de n'étendre que les portées de noms qui doivent être étendues pour que la communication puisse avoir lieu. La règle (TE-OUV) est ainsi nommée parce qu'elle *ouvre* la portée d'un nom b apparaissant dans le système $(\mathsf{new}\, b : \mathsf{E})\, M$ en supprimant le lieur $(\mathsf{new}\, b : \mathsf{E})$ dès que le système M est capable d'émettre un message mentionnant b. Bien entendu, il faut que le nom ne soit pas celui de la localité ou du canal sur lequel M émet un message, sans quoi $(\mathsf{new}\, b : \mathsf{E})\, M$ serait évidemment incapable d'interagir ainsi avec son environnement. Le nom b ainsi que le type qui lui est associé sont alors rajoutés à l'environnement Φ de l'étiquette. Symétriquement, la règle (TE-COMM) peut réintroduire des lieurs dans le système supprimés par la règle (TE-OUV) : comme l'exemple précédent le montre, toutes les portées des noms qui doivent être étendues pour autoriser la communication n'ont à englober que les deux processus qui communiquent ; elles peuvent par conséquent être toutes refermées au niveau de la composition parallèle qui réunit les deux protagonistes.

La règle (TE-C-NEW) s'oppose à (TE-OUV), car elle ignore tout simplement les lieurs lorsqu'ils portent sur des noms non mentionnés dans l'étiquette : $\mathsf{n}(\mu)$ désigne l'ensemble de tous les noms de l'étiquette μ, qu'ils y soient liés (dans le domaine de l'environnement Φ si cette étiquette est de la forme $(\Phi)a!V$ ou $(\Phi)a?V$) ou libre (dans V ou dans Φ). Les conditions de la règle (TE-OUV), quant à elle, ne mentionnent que les noms de V car il est impossible, lors d'une transition étiquetée par $(\Phi)a!V$, qu'un nom apparaisse dans les types de Φ sans être mentionné dans V. Le système

$$(\mathsf{new}\, k : \textsc{loc})\, (\mathsf{new}\, c : \textsc{rw}\langle\rangle_{@} k)\, l[\![a\,!\,\langle c\rangle]\!]$$

permet d'illustrer pourquoi tous les noms qui doivent être ouverts apparaissent nécessairement dans la valeur transmise. Ce système pourrait essayer d'effectuer une transition étiquetée par $(c : \textsc{rw}\langle\rangle_{@} k)a!c$, où k est mentionnée sans être ouverte. Mais un tel système ne peut pas être bien typé car a devrait avoir le

[6] Les étiquettes présentées ici sont atypiques dans le monde du Dπ-calcul. Elles mentionnent habituellement la localité où l'action a lieu. Cette information n'est cependant pas nécessaire ici car les transitions sont définies uniquement dans un cadre typé où l'environnement de l'observateur (Ω) associe toujours une localité au canal sur lequel l'action est effectuée. La localité peut donc être omise dans l'étiquette puisqu'elle est dans Ω.

type $\mathrm{W}\langle\mathrm{RW}\langle\rangle_{@}k\rangle$ qui est mal formé : $\mathrm{RW}\langle\rangle_{@}k$ n'est en effet pas totalement clos. Ce constat se généralise facilement à toute transition.

Suivant l'intuition déjà expliquée, les règles (TE-R) et (TE-W) supposent que l'observateur dispose de la capacité duale de celle utilisée par le système observé, puisque ces règles identifient les cas de possibles interactions entre l'environnement et le système : pour interagir avec une émission, il faut être capable de recevoir le message, et vice versa. La règle (TE-R) vérifie aussi que le message émis par l'observateur et le canal sur lequel l'émission a lieu ont des types cohérents pour autant que sache l'observateur. La bonne formation de la configuration $\Omega \rhd l[\![a\,?\,(X : \mathsf{T})\,P]\!]$ assurera alors que le type T' auquel le message est émis est effectivement un sous-type de T. On sait en effet qu'il existe un environnement Γ sous-type de Ω et tel que $\Gamma \vdash l[\![a\,?\,(X : \mathsf{T})\,P]\!]$ donc permettant de prouver $\Gamma \vdash a : \mathrm{R}\langle\mathsf{T}\rangle_{@}l$. Puisque Γ est un sous-type de Ω dans lequel le type $\mathrm{W}\langle\mathsf{T}'\rangle_{@}l$ est associé à a, il doit aussi être possible de prouver $\Gamma \vdash a : \mathrm{W}\langle\mathsf{T}'\rangle_{@}l$. Ces deux jugements assurent que $\mathrm{RW}\langle\mathsf{T},\mathsf{T}'\rangle_{@}l$ est un type donc que T' est un sous-type de T.

Un raisonnement reposant également sur l'environnement Γ qui sous-tend la configuration $\Omega \rhd l[\![a\,!\,\langle V\rangle\,P]\!]$ de (TE-W) garantit que $\Omega, \langle V : \mathsf{T}\rangle_{@}l$ soit bien formé : en effet $\Gamma <: \Omega$ et $\Gamma <: \langle V : \mathsf{T}\rangle_{@}l$ ce qui assure les conditions de compatibilité quand les hypothèses de $\langle V : \mathsf{T}\rangle_{@}l$ sont rajoutées à Ω. Par ailleurs, la règle (TE-W) emploie la notation $\Omega^r(a)$ pour désigner la borne inférieure des types auxquels sont reçues les valeurs sur le canal a pour l'environnement Ω, c'est-à-dire

$$\Omega^r(a) = \bigsqcap\{\mathsf{T} \mid \exists s, a : \mathrm{R}\langle\mathsf{T}\rangle_{@}s \in \Omega \text{ ou } \exists s, \exists \mathsf{T}', a : \mathrm{RW}\langle\mathsf{T},\mathsf{T}'\rangle_{@}s \in \Omega\}$$

Ce type est bien entendu défini parce que la contrainte de bonne formation de l'environnement Ω impose que tous les types associés à a soient compatibles (au sens large mais aussi au sens strict puisque les types dans Ω ne contiennent que des noms) et la règle (TE-W) assure que la capacité de réception doit y être disponible. Le théorème 2.17 garantit par conséquent que cet ensemble non vide de types admet une borne inférieure.

Enfin, la règle (TE-AFF) autorise l'observateur à introduire des noms inédits lorsqu'il émet un message à destination d'un système. Les autres règles sont assez naturelles.

Bien que l'objectif du STTE ne soit pas de fournir la sémantique du calcul, puisqu'elle a déjà été formalisée dans un cadre bien plus simple et non-typé au chapitre précédent, la cohérence de cette définition ne peut se prouver que par une adéquation de ces deux sémantiques.

Théorème 2.37 (Coïncidence des sémantiques)**.** *Les réductions du système typé de transitions étiquetées et de la sémantique par réductions coïncident au sens des deux propriétés suivantes :*

- $\Omega \rhd M \xrightarrow{\tau} \Omega \rhd M'$ *implique que* $M \longrightarrow M'$ *;*
- $M \longrightarrow M'$ *et la bonne formation de* $\Omega \rhd M$ *impliquent l'existence d'un* M'' *tel que* $M'' \equiv M'$ *et* $\Omega \rhd M \xrightarrow{\tau} \Omega \rhd M''$*.*

Ce théorème permet donc de choisir l'une ou l'autre des deux sémantiques dans les démonstrations, le STTE garantissant une relative conservation de la structure des systèmes puisque la congruence structurelle en est bannie. La preuve de ce théorème sera décomposée au chapitre 4 où les différents lemmes intermédiaires seront particulièrement utiles. Évoquons simplement ici un lemme qui servira au chapitre 3.

Lemme 2.38 (Compatibilité entre la congruence structurelle et le STTE)**.**

$$\text{Si}\quad \begin{array}{ccc} \Omega \rhd M_1 & \equiv_t & \Omega \rhd M^1 \\ {\scriptstyle\mu}\downarrow & & \\ \Omega' \rhd M_2 & & \end{array} \qquad \text{alors}\quad \begin{array}{ccc} \Omega \rhd M_1 & \equiv_t & \Omega \rhd M^1 \\ {\scriptstyle\mu}\downarrow & & \downarrow{\scriptstyle\bar{\mu}} \\ \Omega' \rhd M_2 & \equiv_t & \Omega' \rhd M^2 \end{array}$$

où $\bar{\mu}$ est
- *μ si $\mu = \tau$ ou $\mu = (\Phi)a?V$;*
- *$(\Phi)a!V$ si $\mu = (\Phi')a!V$ où Φ et Φ' sont le même environnement à une permutation des hypothèses près.*

Puisque le STTE a été introduit afin de fournir une notion plus manipulable d'équivalence sur les systèmes, nous allons maintenant formaliser cette équivalence appelée *bisimulation*. Cette équivalence repose sur le principe que deux systèmes, pouvant interagir de façon indiscernable avec des contextes « bien choisis », seront indiscernables quels que soient les contextes dans lesquels ils seront placés. C'est pourquoi les étiquettes du STTE correspondent intuitivement à des contextes. Mais elles contiennent aussi certaines informations pour que la sémantique engendrée par le STTE soit la « bonne », c'est-à-dire qu'elle coïncide avec la sémantique par réductions. Considérons par exemple la configuration

$$l : \text{LOC}, a : \text{RW}\langle \text{R}\langle\rangle\rangle_{@}l \rhd (\mathsf{new}\, c : \text{RW}\langle\rangle_{@}l)\, l[\![a\,!\,\langle c\rangle]\!]$$

qui peut effectuer, suivant les règles de la figure 2.13, une transition $\xrightarrow{(c:\text{RW}\langle\rangle@l)a!c}$. Malheureusement, cette étiquette donne bien plus d'informations que l'observateur ne peut en voir : seule la capacité de réception sur le nom c est visible pour l'observateur car le type auquel il connaît a est $\text{RW}\langle \text{R}\langle\rangle\rangle_{@}l$. Par conséquent, cet observateur est incapable de faire la différence entre ce système et

$$l : \text{LOC}, a : \text{RW}\langle \text{R}\langle\rangle\rangle_{@}l \rhd (\mathsf{new}\, c : \text{R}\langle\rangle_{@}l)\, l[\![a\,!\,\langle c\rangle]\!]$$

bien que ce dernier effectue une transition $\xrightarrow{(c:\text{R}\langle\rangle@l)a!c}$.

Pour éviter de différencier ces deux interactions, on définit les *actions* qui mentionnent sensiblement moins d'informations que les étiquettes des transitions.

Définition 2.39 (Actions)**.** *On dit que la configuration $\Omega \rhd M$ peut accomplir l'action μ, et on note $\Omega \rhd M \xrightarrow{\mu}$, si :*
- *on peut prouver $\Omega \rhd M \xrightarrow{\mu} \Omega' \rhd M'$ dans le système typé de transitions étiquetées quand μ est de la forme τ ou $(\Phi)a?V$;*
- *on peut prouver $\Omega \rhd M \xrightarrow{(\Phi)a!V} \Omega' \rhd M'$ dans le système typé de transitions étiquetées quand μ est de la forme $(\tilde{b})a!V$ où $\tilde{b} = \mathsf{dom}(\Phi)$.*

Les notations pour les actions et les transitions sont confondues, on lèvera le doute qui pourrait exister dans les cas où la différence importe dans la suite.

Cette amputation de l'étiquette n'empêche pas de conserver une propriété fondamentale : quelle que soit l'action μ effectuée par une configuration $\Omega \rhd M$, l'environnement Ω' représentant la connaissance de l'observateur après l'interaction ne dépend que de Ω et de μ.

Définition 2.40 (Environnement résultant)**.** *Quand une configuration $\Omega \rhd M$ effectue une action μ, on appelle* environnement résultant *l'environnement, noté « Ω après μ » :*

- $\Omega, \langle V : \Omega^r(a) \rangle \circledast l$ *quand* μ *est l'action* $(\tilde{b})a!V$;
- $\Omega; \Phi$ *quand* μ *est l'action* $(\Phi)a?V$;
- Ω *quand* $\mu = \tau$.

Une très simple induction sur le STTE permet de voir que, dès que la configuration $\Omega \rhd M$ peut faire une action μ, la configuration obtenue sera toujours de la forme Ω après $\mu \rhd M'$.

Exactement comme les réductions $\longrightarrow$ ont une variante *faible*, $\Longrightarrow$, les actions $\xrightarrow{\mu}$ peuvent être affaiblies.

Définition 2.41 (Actions faibles)**.** *On appelle* action faible*, notée* $\xRightarrow{\hat{\mu}}$ *:*

- $\xrightarrow{\tau}{}^*$ *si* $\mu = \tau$;
- $\xrightarrow{\tau}{}^* \xrightarrow{\mu} \xrightarrow{\tau}{}^*$ *sinon.*

Maintenant que les informations excessives apparaissant dans les étiquettes sont effacées, définissons la notion de bisimulation qui découle du STTE.

Définition 2.42 (Bisimulation typée)**.** *La relation typée* $\mathcal{R}$ *est une* bisimulation typée *si* $\Omega \vDash M \mathcal{R} N$ *implique :*

- *si* $\Omega \rhd M \xrightarrow{\mu} \Omega$ après $\mu \rhd M'$ *alors il doit exister un système* N' *tel que* $\Omega \rhd N \xRightarrow{\hat{\mu}} \Omega$ après $\mu \rhd N'$ *et* Ω après $\mu \vDash M' \mathcal{R} N'$;
- *symétriquement, si* $\Omega \rhd N \xrightarrow{\mu} \Omega$ après $\mu \rhd N'$ *alors il doit exister un système* M' *tel que* $\Omega \rhd M \xRightarrow{\hat{\mu}} \Omega$ après $\mu \rhd M'$ *et* Ω après $\mu \vDash M' \mathcal{R} N'$.

La définition des relations de bisimulation correspond à une intuition simple. On considère en effet que l'ensemble des actions a été suffisamment bien choisi pour caractériser toutes les interactions que le système peut engager avec son environnement. Par conséquent, si une relation est une bisimulation typée et contient un triplet (Ω, M, N), on sait alors que, quelles que soient les suites d'actions que les systèmes M et N effectueront, tous les états qu'ils pourront atteindre auront un équivalent pour l'autre système. En poursuivant l'intuition qui avait conduit à la définition de la congruence typée barbue fermée par réductions, ces deux systèmes seront indiscernables pour un observateur ne connaissant que Ω puisqu'ils pourront toujours effectuer les mêmes interactions que l'autre, quelle qu'ait été la suite d'interactions préalables.

Par ailleurs, les bisimulations typées s'organisent de façon très structurée. En effet, on se convainc aisément qu'une union quelconque de bisimulations est toujours une bisimulation. Par conséquent, il est possible de considérer l'union de toutes les bisimulations, pour obtenir l'équivalence la plus large qui respecte la définition des bisimulations.

Définition 2.43 (Bisimilarité typée)**.** *On appelle* bisimilarité typée *la plus grande bisimulation typée. On la note* $\approx^t$.

La définition de la bisimilarité autorise l'utilisation de la même méthodologie que la congruence barbue fermée par réductions pour prouver que deux systèmes sont équivalents pour une certaine connaissance du système. On pourra ainsi exhiber une bisimulation typée qui contient ces deux systèmes avec l'environnement représentant l'observateur. La preuve s'arrêtera là car cette bisimulation typée sera nécessairement incluse dans la bisimilarité typée.

Voyons donc un exemple de cette puissante technique de preuve d'équivalence que la bisimulation autorise.

Exemple 2.44. Reprenons l'exemple 1.14 qui était très simple mais déjà fastidieux pour la congruence barbue fermée par réductions et démontrons que les deux systèmes $M = (\mathsf{new}\, a : \textsc{rw}\langle\rangle@l)\, l[\![a\,!\,\langle\rangle]\!] \mid l[\![a\,?\,()]\!]$ et $\mathbf{0}$ sont équivalents quelles que soient les connaissances de l'observateur, pour peu qu'il ne considère pas le nom l comme un canal.

Pour ce faire, exhibons une bisimulation typée contenant la paire $(M, \mathbf{0})$. En reprenant aussi le système $M' = (\mathsf{new}\, a : \textsc{rw}\langle\rangle@l)\, l[\![\mathsf{stop}]\!] \mid l[\![\mathsf{stop}]\!]$ considérons la relation

$$\mathcal{R} = \{(\Omega, M, \mathbf{0}), (\Omega, M', \mathbf{0}) \mid \Omega \text{ tel que } \Omega \rhd M \text{ soit une configuration}\}$$

La contrainte indiquant que $\Omega \rhd M$ doit être une configuration implique uniquement que si l est dans $\mathsf{dom}(\Omega)$, alors le type qui lui est associé doit être LOC. Or la seule action que les configurations $\Omega \rhd M$ peuvent faire est $\Omega \rhd M \xrightarrow{\tau} \Omega \rhd M'$. La relation $\mathcal{R}$ vérifie alors les conditions d'une bisimulation grâce à l'action faible $\Omega \rhd \mathbf{0} \stackrel{\hat{\tau}}{\Longrightarrow} \Omega \rhd \mathbf{0}$ puisque l'action faible $\stackrel{\hat{\tau}}{\Longrightarrow}$ est une suite d'un nombre arbitraire, zéro compris, d'actions étiquetées τ. Cette action faible convient car $\Omega \vDash M' \,\mathcal{R}\, \mathbf{0}$ est effectivement dans la relation. Les trois autres conditions qui sont à vérifier expriment le fait qu'à toute action de $\Omega \rhd \mathbf{0}$, $\Omega \rhd M'$ et $\Omega \rhd \mathbf{0}$ doit correspondre une action faible de, respectivement, $\Omega \rhd M$, $\Omega \rhd \mathbf{0}$ et $\Omega \rhd M'$. Or ces trois premières configurations ne peuvent pas effectuer la moindre action, ces trois conditions sont donc immédiatement vérifiées. Cela achève la preuve que la « petite » relation $\mathcal{R}$ est une bisimulation typée et, par conséquent, que $\Omega \vDash M \approx^t \mathbf{0}$ pour tout Ω tel que $\Omega \rhd M$ soit une configuration.

Nous avons donné deux équivalences typées : la congruence typée barbue fermée par réductions, dûment justifiée comme étant l'équivalence observationnelle typée que l'intuition dicte ; et la bisimulation typée qui fournit une méthode permettant de réellement prouver des équivalences de systèmes placés dans certains environnements. Le résultat qui manque est bien entendu :

Théorème 2.45 (Coïncidence des équivalences typées)**.** $\cong^{tbr} = \approx^t$

Encore une fois, ce théorème se prouve de façon très similaire à la preuve qui apparaîtra au chapitre 4, on ne donne pas ici les nombreux détails qui jalonnent cette preuve.

Chapitre 3

Implémentation de la récursion par réplication

3.1 Liens entre la réplication et la récursion

Le Dπ-calcul que nous avons présenté dans les chapitres précédents fournit deux mécanismes permettant de modéliser des comportements infinis : les processus récursifs de la forme $(\mathsf{rec}\ Z : \mathsf{R}(X)\,.\,P)\,\langle V\rangle$ et les processus répliqués de la forme $*P$. Alors que toutes les autres constructions de processus remplissent des offices très différents, on imagine assez aisément qu'il existe certains comportements de processus que l'on modélisera aussi facilement avec l'une ou l'autre. Pour prendre un exemple d'une extrême simplicité, un processus émettant sans fin sur le canal c de la localité l le nom de la localité k pourra s'écrire aussi bien $l[\![*c\,!\,\langle k\rangle]\!]$ que $l[\![(\mathsf{rec}\ Z : \mathsf{R}(x)\,.\,x\,!\,\langle k\rangle \mid Z\,\langle x\rangle)\,\langle c\rangle]\!]$ avec $\mathsf{R} = \sum y : \textsc{loc}.\ \mathrm{W}\langle\textsc{loc}\rangle_{@y}$.

Ce petit exemple illustre un aspect essentiel du lien existant entre ces deux constructions : il est possible d'exprimer tout comportement infini avec l'une ou l'autre mais leur formulation correspond naturellement à des cas très différents. L'exemple montre ainsi qu'exprimer un processus naturellement répliqué à l'aide d'une récursion demande une paramétrisation sur les noms des canaux locaux utilisés par le processus. En effet, le typage mis en place au chapitre 2 empêche les processus récursifs d'utiliser des canaux dans la localité où ils sont déclenchés si ces canaux ne sont pas reçus en argument. Il ne fait pas d'exception pour les cas où toutes les instances du processus récursif sont déclenchées dans la même localité, cas particuliers où les précautions du typage s'avèrent excessives.

Cette grande différence au typage est la conséquence de la principale distinction que l'on puisse faire entre la récursion et la réplication dans un cadre réparti comme celui du Dπ-calcul. En effet, toutes les itérations d'un processus répliqué $l[\![*P]\!]$ seront lancées dans la même localité l tandis que les diverses instanciations du processus récursif $l[\![(\mathsf{rec}\ Z : \mathsf{R}(X)\,.\,P)\,\langle V\rangle]\!]$ pourront bien entendu démarrer dans d'autres localités si le processus P en décide ainsi. Les processus récursifs peuvent ainsi réellement modéliser des agents qui explorent leur environnement en migrant de proche en proche, comme le prospecteur présenté au chapitre 1. Cette grande différence entre réplication et récursion justifie la conservation des deux constructions dans le Dπ-calcul. Cependant cette différence n'a pas de sens dans les calculs non répartis, notamment le π-calcul où, généralement, seule la réplication est utilisée parce qu'elle est plus aisée à contrôler (voir par exemple [SW01]).

Malgré leurs différences, il est possible d'implémenter les réplications par des récursions comme on l'a esquissé sur le petit exemple précédent, et, réciproquement, les récursions par des réplications. Cependant, cette dernière aura un coût élevé en nombre de migrations de processus, puisque les différentes instances d'un processus répliqué sont toujours déclenchées dans une même localité. Cela implique de plus que, si le formalisme prenait en compte d'éventuelles pannes (voir par exemple [FH05]), l'implémentation d'un processus récursif sous forme de processus répliqué aurait d'autant plus de risque de se confronter à une telle panne. On conjecture qu'il serait alors impossible d'obtenir une implémentation qui soit équivalente, au sens de la congruence barbue typée définie au chapitre 2, au processus original.

Puisque les processus répliqués sont ancrés dans une localité, on associera une *localité-refuge* à chaque processus répliqué mimant un processus récursif.

Illustrons comment le prospecteur, récursif, peut être décrit par un processus répliqué et des migrations. Rappelons tout d'abord le prospecteur.

$$
\begin{aligned}
l\ [\![\ & (\mathsf{rec}\ Z : \mathsf{R}\,(x_{\mathrm{val}}, x_{\mathit{vois}})\,.\\
&\quad x_{\mathit{val}}\,?\,(x : \mathsf{T})\\
&\quad \mathsf{if}\ x = \mathit{or}\\
&\quad \mathsf{then\ here}\,(y)\,\mathsf{goto}\ \mathit{campement}.\ \mathit{locOr}\,!\,\langle y\rangle\\
&\quad \mathsf{else}\ x_{\mathit{vois}}\,?\,((y, X) : \mathsf{T}')\,\mathsf{goto}\ y.\ Z\,\langle X\rangle\\
&)\,\langle \mathit{val}, \mathit{vois}\rangle \qquad]\!]
\end{aligned}
$$

On peut alors proposer une version répliquée qui effectue le même calcul, de la forme suivante :

$$\mathit{ref}_Z\,[\![*\mathsf{ProspectIter}]\!] \mid l[\![\mathsf{DéclenchInst}_{\mathit{val},\mathit{vois}}]\!]$$

avec

$$
\begin{aligned}
\mathsf{ProspectIter} \quad &\triangleq \quad \mathit{inst}_Z\,?\,((x, (x_{\mathrm{val}}, x_{\mathrm{vois}})) : \mathsf{R})\,\mathsf{goto}\ x.\\
&\qquad x_{\mathit{val}}\,?\,(x : \mathsf{T})\\
&\qquad \mathsf{if}\ x = \mathit{or}\\
&\qquad \mathsf{then\ here}\,(y)\,\mathsf{goto}\ \mathit{campement}.\ \mathit{locOr}\,!\,\langle y\rangle\\
&\qquad \mathsf{else}\ x_{\mathit{vois}}\,?\,((y, X) : \mathsf{T}')\,\mathsf{goto}\ y.\ \mathsf{DéclenchInst}_X\\
\mathsf{DéclenchInst}_V \quad &\triangleq \quad \mathsf{here}\,(x)\,\mathsf{goto}\ \mathit{ref}_Z.\ \mathit{inst}_Z\,!\,\langle x, V\rangle
\end{aligned}
$$

et

$$\mathsf{R} = \textsc{rec}\,\mathbf{Y}.\ \sum x : \textsc{loc}.\ (\mathsf{C}_{@}x, \textsc{r}\langle\mathbf{Y}\rangle_{@}x)$$

Le fonctionnement de cette traduction est très simple. Seules les constructions de la récursion sont touchées. Ainsi les appels récursifs deviennent de petits processus qui se chargent de migrer dans le refuge du processus répliqué et de lui transmettre la localité de l'appel récursif et les arguments de cet appel. La traduction de la déclaration du processus récursif est plus complexe :

- elle produit un processus répliqué ;
- ce processus répliqué commence par recevoir une paire comprenant la localité où une instance du processus est invoquée et le jeu d'arguments correspondant à cette invocation ;
- il migre ensuite dans la localité en question pour exécuter la nouvelle instance du processus ;

– par ailleurs la traduction commence par déclencher une première instance du processus récursif dans la localité où le processus récursif est déclaré.

Cette traduction permet que le *corps* du processus récursif, c'est-à-dire P dans $(\mathsf{rec}\ Z : \mathsf{R}\,(X)\,.\,P)\,\langle V\rangle$, devienne un processus répliqué sans modification.

Il saute aux yeux que la traduction de ce processus répliqué pourrait être optimisée en transformant $\mathsf{goto}\,y.\,\mathsf{DéclenchInst}_X$ de sorte à éviter une migration vers la localité y uniquement pour y récupérer le nom y et le transmettre au processus répliqué sur le canal $inst_Z$. Ce morceau du processus pourrait donc se récrire directement sous la forme $\mathsf{goto}\,ref_Z.\,inst_Z\,!\,\langle y, V\rangle$.

Cette optimisation n'est cependant possible que dans la mesure où l'appel récursif est effectué immédiatement après la migration vers la nouvelle localité où l'exploration se poursuit. La première traduction que l'on a donnée, en revanche, ne dépend aucunement de la façon dont le processus récursif proprement dit est écrit. C'est par conséquent suivant cette piste que nous allons définir une traduction systématique des processus récursifs en processus répliqués.

3.2 Description de l'implémentation

Définissons donc une traduction générale des processus récursifs en utilisant les idées présentées sur l'exemple du prospecteur. Une question assez délicate a été laissée de côté dans cet exemple : quand est-ce que la localité ref_Z servant de refuge à un processus doit-elle être introduite ? Les deux réponses possibles à une telle question sont :

– introduire la localité ref_Z et y placer le processus répliqué lors de la traduction,
– ou bien donner une traduction qui génère la localité et le processus répliqué qu'elle contient à l'instant où ils sont nécessaires.

Cette question peut paraître anecdotique dans l'exemple précédent, parce que le processus récursif dans $l[\![(\mathsf{rec}\ Z\,(X)\,.\,\cdots)\,\langle V\rangle]\!]$ n'est pas placé sous un préfixe. Pourtant le prospecteur utilise des noms, comme *base* ou *locOr*, qui ne sont pas liés dans les paramètres du processus. Si, en omettant les informations de type, on considère le système $(\mathsf{new}\,base)\,(\mathsf{new}\,locOr)\,l[\![\mathsf{Prospecteur}]\!]$, on voit alors que la traduction ne pourra pas prendre la forme

$$ref_Z[\![*\mathsf{ProspectIter}]\!]\,|\,(\mathsf{new}\,base)\,(\mathsf{new}\,locOr)\,l[\![\mathsf{DéclenchInst}_V]\!]$$

La première solution consiste à mettre en place toutes les localités-refuges lors de la traduction. Cela signifie que la traduction d'un système complet M commence alors par extraire l'ensemble des processus récursifs qu'il contient. En les prenant dans un ordre arbitraire, on notera le i^{e} processus récursif $\mathsf{rec}\ Z_i : \mathsf{R}_i\,(X_i)\,.\,P_i$. Pour tourner des processus récursifs sous forme répliquée, il faut remplacer tous les identifiants libres dans les processus récursifs par des variables ajoutées à l'ensemble des valeurs reçues sur le canal $inst_{Z_i}$ lorsqu'une nouvelle instance est déclenchée. En notant $\tilde{u}_i$ pour $\mathsf{fn}(P_i) \cup \mathsf{fv}(P_i)$, c'est-à-dire l'ensemble des identifiants libres de P_i, la traduction des systèmes et processus se fait par induction structurelle et prend la forme suivante :

$$\text{D-NC}(Z_i\,\langle V\rangle) = \mathsf{here}\,(x)\,\mathsf{goto}\,ref_{Z_i}.\,inst_{Z_i}\,!\,\langle x, V, \vec{u}_i\rangle$$
$$\text{D-NC}((\mathsf{rec}\ Z_i : \mathsf{R}_i\,(X_i)\,.\,P_i)\,\langle V\rangle) = \text{D-NC}(Z_i\,\langle V\rangle)$$

et la traduction passe directement sur la ou les continuations pour toutes les autres constructions. Les traductions des appels récursifs et des déclarations des processus récursifs sont identiques puisque les localités-refuges et les processus répliqués sont ajoutés par ailleurs. Pour cela, on crée les localités-refuges ref_{Z_i} une fois pour toute et on y place le processus répliqué correspondant à P_i. Ces localités-refuges et leur processus répliqué sont positionnés en tête de la traduction, ce qui donne une traduction complète du système M qui ressemble donc, en omettant les informations de types, à :

$$\begin{array}{l}(\mathsf{new}\, ref_{Z_1})\,(\mathsf{new}\, inst_{Z_1})\,(\mathsf{new}\, ref_{Z_2})\,(\mathsf{new}\, inst_{Z_2})\cdots \\ \quad ref_{Z_1}[\![*inst_{Z_1}\,?\,(x, X_1, Y_1)\,\mathsf{goto}\, x.\ \text{D-NC}(P_1)\{^{Y_1}/_{\vec{u}_1}\}]\!]\,| \\ \quad ref_{Z_2}[\![*inst_{Z_2}\,?\,(x, X_2, Y_2)\,\mathsf{goto}\, x.\ \text{D-NC}(P_2)\{^{Y_2}/_{\vec{u}_2}\}]\!]\,|\cdots| \\ \quad \text{D-NC}(M)\end{array}$$

Cette traduction est très proche de celle standard pour le π-calcul (présentée notamment dans [SW01]). Elle n'est, en revanche, absolument pas compositionnelle, contrairement à la seconde solution proposée. On adoptera donc ici cette seconde voie : la traduction générera les localités-refuges à la volée quand elles sont requises. On notera $(\!|M|\!)$ la traduction du système M et $(\!|P|\!)$ celle du processus P. On suppose naturellement que toutes les variables de récursion sont distinctes et que les noms ref_Z et $inst_Z$ et la variable x_Z sont inédits dans le système ou le processus à traduire. La traduction des deux constructions pour les processus récursifs est alors la suivante :

$$\begin{array}{l}(\!|Z\,\langle V\rangle|\!) = \mathsf{here}\,(x_Z)\,\mathsf{goto}\, ref_Z.\ inst_Z\,!\,\langle x_Z, V\rangle \\ (\!|(\mathsf{rec}\ Z : \mathsf{R}\,(X)\,.\,P)\,\langle V\rangle|\!) = \\ \qquad \mathsf{newloc}\, ref_Z, inst_Z : \sum x : \text{LOC}.\ \text{RW}\langle \mathsf{R}\rangle_{@x}\, \mathsf{with} \\ \qquad\qquad *inst_Z\,?\,((x_Z, X) : \mathsf{R})\,\mathsf{goto}\, x_Z.\ (\!|P|\!)\ \mathsf{in} (\!|Z\,\langle V\rangle|\!)\end{array}$$

Remarque 3.1. Dans cette traduction, la remarque 2.22 (page 67) à propos des types R devient primordiale : cette remarque dit que les types de canaux C non localisés mentionnés dans $\mathsf{R} = \sum x : \text{LOC}.\ \mathsf{T}$ peuvent aussi être explicitement localisés sous la forme $\mathsf{C}_{@x}$, sans changer le typage des processus. Pour que la traduction fournie ci-dessus soit valable, il est indispensable que tous les types de canaux soient explicitement localisés car R est utilisé pour typer des valeurs transmises au sein de la localité ref_Z tandis que les canaux transmis doivent être situés dans la localité x_Z.

Remarque 3.2. Un autre point capital de cette traduction est le fait que le type R est conservé intact. Or les types récursifs ont été introduits au chapitre 2 en considérant que le typage du prospecteur nécessitait de tels types. Cette traduction met en lumière que les types récursifs sont indispensables pour typer tous les *comportements récursifs*, qu'ils soient exprimés en utilisant la récursion ou bien la réplication.

La fonction $(\!|\cdot|\!)$ appliquée à une autre construction de processus passe directement sur la ou les continuations :

$$\begin{array}{l}(\!|u\,!\,\langle V\rangle\, P|\!) = u\,!\,\langle V\rangle\,(\!|P|\!) \\ (\!|u\,?\,(X : \mathsf{T})\, P|\!) = u\,?\,(X : \mathsf{T})\,(\!|P|\!)\end{array}$$

et ainsi de suite. Il en va de même pour toutes les constructions de système :

$$\begin{aligned}
(\!|\mathbf{0}|\!) &= \mathbf{0} \\
(\!|l[\![P]\!]|\!) &= l[\![(\!|P|\!)]\!] \\
(\!|M_1 \mid M_2|\!) &= (\!|M_1|\!) \mid (\!|M_2|\!) \\
(\!|(\mathsf{new}\, a : \mathsf{E})\, M|\!) &= (\mathsf{new}\, a : \mathsf{E})(\!|M|\!)
\end{aligned}$$

Cette seconde traduction présente donc les avantages d'être compositionnelle et de générer les localités-refuges à la demande, donc uniquement si elles s'avèrent réellement utiles. Par ailleurs, elle est plus élégante dans la mesure où elle ne requiert aucune modification portant sur les noms et variables libres du processus récursifs.

3.3 Équivalence d'un processus récursif et de son traduit

3.3.1 Difficultés à aborder pour l'équivalence

La fonction $(\!|\cdot|\!)$ décrite à la section précédente transforme tout système M en un système qui ne contient plus de processus récursif. Pour prouver qu'il s'agit bien là d'une traduction, nous allons maintenant montrer que le système obtenu est équivalent à celui de départ, en utilisant la notion d'équivalence définie au chapitre précédent : la congruence typée barbue fermée par réduction (définition 2.35). Puisque cette équivalence est typée, le premier résultat à montrer est le fait que les systèmes M et $(\!|M|\!)$ sont bien typés dans les mêmes environnements. Cette propriété nous permettra de nous fixer comme objectif la preuve de $\Omega \vDash M \cong^{tbr} (\!|M|\!)$ si Ω est un environnement tel que $\Omega \rhd M$ soit une configuration bien formée. D'après le théorème 2.45 de coïncidence des équivalences typées, il suffit de montrer que les systèmes M et $(\!|M|\!)$ sont bisimilaires dans Ω pour obtenir ce résultat.

Pour prouver que ces deux systèmes sont effectivement bisimilaires, il nous faut exhiber une relation qui les contienne tous les deux et qui soit incluse dans la bisimilarité. Malheureusement, les bisimulations qui contiennent à la fois les systèmes M et $(\!|M|\!)$ sont relativement complexes. Si on considère l'exemple très simple

$$l[\![(\mathsf{rec}\ Z : \mathsf{R}\,(x)\,.\ x\,!\,\langle\rangle\ Z\,\langle x\rangle)\,\langle a\rangle]\!]$$

sa traduction est

$$\begin{aligned}
l[\![&\mathsf{newloc}\, ref_Z, inst_Z : \textstyle\sum x : \text{LOC. } \text{RW}\langle \mathsf{R}\rangle @ x \ \mathsf{with} \\
&\quad *inst_Z\,?\,((x_Z, x) : \mathsf{R})\ \mathsf{goto}\ x_Z\,.\ x\,!\,\langle\rangle\ (\!|Z\,\langle x\rangle|\!)\ \mathsf{in} (\!|Z\,\langle a\rangle|\!)]\!]
\end{aligned}$$

avec $(\!|Z\,\langle x\rangle|\!) = \mathsf{here}\,(y)\ \mathsf{goto}\ ref_Z.\ inst_Z\,!\,\langle y, x\rangle$. Cela signifie que lorsque le système effectue l'appel récursif

$$l[\![(\mathsf{rec}\ Z : \mathsf{R}\,(x)\,.\ x\,!\,\langle\rangle\ Z\,\langle x\rangle)\,\langle a\rangle]\!] \xrightarrow{\tau} l[\![a\,!\,\langle\rangle\ (\mathsf{rec}\ Z : \mathsf{R}\,(x)\,.\ x\,!\,\langle\rangle\ Z\,\langle x\rangle)\,\langle a\rangle]\!]$$

sa traduction doit effectuer six transitions étiquetées par τ pour arriver au système :

$$\begin{aligned}
&(\mathsf{new}\, ref_Z)\,(\mathsf{new}\, inst_Z) \\
&\qquad ref_Z[\![*inst_Z\,?\,((x_Z, x) : \mathsf{R})\ \mathsf{goto}\ x_Z\,.\ x\,!\,\langle\rangle\ (\!|Z\,\langle x\rangle|\!)]\!] \\
&\quad \mid \quad l[\![a\,!\,\langle\rangle\ (\!|Z\,\langle a\rangle|\!)]\!] \mid ref_Z[\![\mathsf{stop}]\!]
\end{aligned}$$

Pour obtenir une relation de bisimulation contenant les deux systèmes initiaux, il est alors nécessaire de définir une relation contenant aussi toutes les étapes intermédiaires. Cependant, les deux systèmes $l[\![\mathsf{goto}\, ref_Z.\ P]\!]$ et $ref_Z[\![P]\!]$, qui apparaîtront au cours des étapes intermédiaires dues à la traduction, sont indistinguables pour la congruence barbue, puisque l'unique action que puisse effectuer le premier est d'évoluer silencieusement vers le second. Dans de tels cas, il devrait être suffisant de se restreindre à une relation ne contenant que l'un des deux systèmes pour vérifier l'équivalence d'un système et de sa traduction.

Prenons un autre cas, dégénéré et simplifié à l'extrême : le processus récursif inactif $l[\![\mathsf{rec}\ Z.\ \mathsf{stop}]\!]$ où les arguments sont vides et le type omis. La traduction de ce système engendrera naturellement une nouvelle localité refuge pour le processus répliqué correspondant au processus récursif, bien que le processus récursif ne puisse effectuer la moindre itération. Après une étape de réduction, on obtient alors le système $l[\![\mathsf{stop}]\!]$. Pour que la traduction $l[\![\mathsf{rec}\ Z.\ \mathsf{stop}]\!]$ puisse se réduire vers la traduction de $l[\![\mathsf{stop}]\!]$, il faudra qu'elle soit « nettoyée » du processus répliqué inutile qu'elle contient.

Pour traiter les difficultés soulevées dans les deux exemples précédents, on définira une technique de preuve appelée la bisimulation modulo β-réductions et bisimilarité forte et on prouvera sa correction. Cette technique permet de rendre implicites des étapes du raisonnement comme celles que nous venons de voir.

Le premier exemple illustre encore une autre difficulté. Après la première réduction du processus récursif qui aboutit à :

$$l[\![a\,!\,\langle\rangle\,(\mathsf{rec}\ Z : \mathsf{R}\,(x)\,.\,x\,!\,\langle\rangle\ Z\,\langle x\rangle)\,\langle a\rangle]\!]$$

la localité ref_Z a été créée et le processus répliqué correspondant au récursif mis en place. Par conséquent, dans la traduction, le processus récursif complet $(\mathsf{rec}\ Z : \mathsf{R}\,(x)\,.\,x\,!\,\langle\rangle\ Z\,\langle x\rangle)\,\langle a\rangle$ ne correspond plus qu'à un simple appel récursif $(\!|Z\,\langle a\rangle|\!)$. Il faudra donc prendre en compte dans la traduction le fait qu'un certain processus récursif puisse avoir été traduit préalablement.

De façon plus générale, considérons les réductions du système élémentaire $l[\![*\mathsf{rec}\ Z.\ Z \mid Z]\!]$ dans lequel on omet les informations de type par simplicité et les paramètres et arguments car ils sont vides.

$$\begin{array}{ll} & l[\![*\mathsf{rec}\ Z.\ Z \mid Z]\!] \\ \xrightarrow{\tau}{}^{2} & l[\![(\mathsf{rec}\ Z.\ Z \mid Z)^{(1)}]\!] \mid l[\![(\mathsf{rec}\ Z.\ Z \mid Z)^{(2)}]\!] \mid l[\![*\mathsf{rec}\ Z.\ Z \mid Z]\!] \\ \xrightarrow{\tau} & l[\![(\mathsf{rec}\ Z.\ Z \mid Z)^{(1)}]\!] \mid l[\![(\mathsf{rec}\ Z.\ Z \mid Z)^{(3)} \mid (\mathsf{rec}\ Z.\ Z \mid Z)^{(4)}]\!] \mid l[\![*\mathsf{rec}\ Z.\ Z \mid Z]\!] \end{array}$$

Les deux premières actions sont des instanciations du processus répliqué et la troisième un dépliage du processus récursif $(\mathsf{rec}\ Z.\ Z \mid Z)^{(2)}$. On sait alors que les processus récursifs (1) et (2) ne seront pas associés au même processus répliqué par la traduction car le préfixe newloc correspondant à leur refuge sera lui-même répliqué. Puisque le processus récursif (2) devient (3) et (4), on devra avoir au final un processus répliqué pour (1) et un autre pour (3) et (4). La traduction devra donc non seulement prendre en compte si une certaine occurrence d'un processus récursif a déjà été traduite, mais aussi quelles sont les autres occurrences qui découlent de la même traduction (et donc liées au même processus répliqué).

3.3.2 Compatibilité de la traduction et du typage

Commençons par vérifier effectivement que la traduction est compatible avec le typage, condition *sine qua non* pour qu'un système et son traduit appartiennent à une équivalence typée comme la congruence barbue typée.

Proposition 3.3. $\Gamma \vdash M$ *si et seulement si* $\Gamma \vdash (\!|M|\!)$

Démonstration. Cette propriété se prouve de façon similaire à une propriété de substitution, par induction sur M, avec une traduction à la fois des systèmes et des environnements. On étend pour cela la traduction $(\!|\cdot|\!)$ aux environnements. Puisque les variables de récursion sont toutes liées dans le processus, si l'environnement Γ contient une variable Z, on sait que le processus en cours de typage est placé sous une déclaration de la forme $(\mathsf{rec}\ Z : \mathsf{R}\,(X)\,.\,P)\,\langle V\rangle$. Le typage de la traduction pourra alors reposer sur les noms et variables introduits par la traduction de la déclaration de Z. En notant R le type associé à la variable de récursion Z, $(\!|\Gamma|\!)$ est défini par :

$$\begin{array}{rcl} (\!|\Gamma, Z : \textsc{loc}|\!) & = & (\!|\Gamma|\!), \mathit{ref}_Z : \textsc{loc}, \mathit{inst}_Z : \textsc{rw}\langle \mathsf{R}\rangle @ \mathit{ref}_Z, x_Z : \textsc{loc} \\ (\!|\Gamma, u : \mathsf{C} @ Z|\!) & = & (\!|\Gamma|\!), u : \mathsf{C} @ x_Z \\ (\!|\Gamma, u : \mathsf{E}|\!) & = & (\!|\Gamma|\!), u : \mathsf{E} \quad \text{sinon} \end{array}$$

La traduction associe à chaque variable de récursion Z une variable x_Z. On choisira cette association bijective, afin d'éviter tout conflit de noms et on notera σ la substitution $\{^{\vec{x}_Z}/_{\vec{Z}}\}$. On prouve alors en cascade :

1. $\Gamma \vdash \mathbf{env}$ si et seulement si $(\!|\Gamma|\!) \vdash \mathbf{env}$;
2. $\Gamma \vdash s : \mathsf{E}$ si et seulement si $(\!|\Gamma|\!) \vdash (s : \mathsf{E})\sigma$ quand s n'est pas de la forme ref_Z ou inst_Z ;
3. $\Gamma \vdash_s V : \mathsf{T}$ si et seulement si $(\!|\Gamma|\!) \vdash_{s\sigma} (V : \mathsf{T})\sigma$ quand V ne mentionne ni ref_Z ni inst_Z ;
4. avec le Δ collectant le type des variables de récursion dans la portée desquelles on se place, $\Gamma \mid \Delta \vdash_s P$ si et seulement si $(\!|\Gamma|\!) \mid \vdash_{s\sigma} (\!|P|\!)$;
5. $\Gamma \vdash M$ si et seulement si $\Gamma \vdash (\!|M|\!)$;

On obtient ces résultats par induction sur les preuves de leur hypothèse.

1. Le résultat de bonne formation des environnements est immédiat.
2. La définition de $(\!|\Gamma|\!)$ assure que l'hypothèse $s : \mathsf{E}$ apparaît dans Γ si et seulement si $(s : \mathsf{E})\sigma$ apparaît dans $(\!|\Gamma|\!)$. Le résultat pour le typage des identifiants découle alors immédiatement du constat que $\mathsf{E}_1 \sqcap \mathsf{E}_2 <: \mathsf{E}_3$ si et seulement si $(\mathsf{E}_1 \sqcap \mathsf{E}_2 <: \mathsf{E}_3)\sigma$, dont on se convainc aisément car E ne peut prendre que les formes $\textsc{loc}$, $\mathsf{C} @ u$ et $\mathsf{C} @ Z$.
3. Le résultat pour les valeurs s'obtient facilement, en utilisant une propriété standard de commutation des substitutions[1] σ et $\{^{\vec{t}}/_{\vec{x}}\}$ dans le cas de la règle (V-DEP).
4. Tout le cœur du raisonnement doit être effectué pour les processus. On considère alors les différentes formes possibles pour P.

[1] Les propriétés de commutation des substitutions de variables dans des termes avec lieurs sont bien connues : voir par exemple le lemme de substitution dans [Bar84]

$Z\langle V\rangle$ **:** La preuve de $\Gamma \mid \Delta \vdash_s P$ repose alors simplement sur $\Gamma \vdash_s (s, V) : \mathsf{R}$. On en déduit que $(\!|\Gamma|\!) \vdash_{s\sigma} ((s, V) : \mathsf{R})\sigma$. Le fait que le type R n'autorise pas de type de canaux non-localisés C est primordial : on peut en effet en déduire que cette preuve n'utilise jamais la règle (V-CANAL-LOC), et par conséquent que ce jugement ne dépend pas de la localité où il est effectué. $(\!|\Gamma|\!) \vdash_{ref_Z} ((s, V) : \mathsf{R})\sigma$ est donc prouvable. Qui plus est $((s, V) : \mathsf{R})\sigma = (s\sigma, V) : \mathsf{R}$ car R ne contient aucune variable libre et V ne peut syntaxiquement pas mentionner une variable de récursion. On en déduit alors facilement que

$$(\!|\Gamma|\!) \mid \vdash_{s\sigma} \mathsf{here}\,(x)\,\mathsf{goto}\,ref_Z.\ inst_Z\,!\,\langle (x, V)\rangle$$

Réciproquement, ce jugement de typage garantit que $(\!|\Gamma|\!) \vdash_{s\sigma} ((s, V) : \mathsf{R})\sigma$, ce qui suffit pour obtenir $\Gamma \vdash_s (s, V) : \mathsf{R}$ puis $\Gamma \mid \Delta \vdash_s Z\langle V\rangle$ avec le Δ dicté par le contexte dans lequel P est placé.

$(\mathbf{rec}\ Z : \mathsf{R}\,(X)\,.\,P')\,\langle V\rangle$ **:** On utilise le même raisonnement qu'au cas précédent pour le fragment de $(\!|(\mathsf{rec}\ Z : \mathsf{R}\,(X)\,.\,P')\,\langle V\rangle|\!)$ qui sert à déclencher une nouvelle instance. Par ailleurs on voit que :

$$\begin{aligned}(\!|\Gamma;\langle (Z, X) : \mathsf{R}\rangle_{@} Z|\!) = {} & (\!|\Gamma|\!)\,; \\ & ref_Z : \textsc{loc}, \\ & inst_Z : \textsc{rw}\langle \mathsf{R}\rangle_{@} ref_Z, \\ & \langle (x_Z, X) : \mathsf{R}\rangle_{@} ref_Z\end{aligned}$$

par un simple raisonnement sur l'expansion $\langle (Z, X) : \mathsf{R}\rangle_{@} Z$ sachant que R ne dépend pas de la localité où il est expansé. L'hypothèse d'induction donne alors

$$\Gamma;\langle (Z, X) : \mathsf{R}\rangle_{@} Z \mid \Delta; Z : \mathsf{R} \vdash_Z P'$$

si et seulement si

$$(\!|\Gamma;\langle (Z, X) : \mathsf{R}\rangle_{@} Z|\!) \mid \vdash_{x_Z} (\!|P'|\!)$$

d'où l'on déduit rapidement le résultat.

$u\,?\,(X : \mathsf{T})\,P'$ **:** On utilise directement les hypothèses d'induction après avoir remarqué que $(\!|\Gamma;\langle X : \mathsf{T}\rangle_{@} s|\!)$ est égal $(\!|\Gamma|\!);\langle X : \mathsf{T}\rangle_{@} s\sigma$ car le type T est clos et les variables du motif X sont liées dans ce processus.

$\mathbf{newchan}\,c : \mathsf{C}\,\mathbf{in}\,P'$ $\mathbf{et\ newloc}\,k, (\vec{c}) : \mathsf{L}\,\mathbf{with}\,P_k\,\mathbf{in}\,P'$ **:** On raisonne de façon similaire au cas précédent.

$\mathbf{if}\ u_1 = u_2\ \mathbf{then}\ P_1\ \mathbf{else}\ P_2$ **:** L'hypothèse d'induction suffit pour le processus P_2. Pour P_1, on remarque que $[(\!|\Gamma|\!)]_{u_1=u_2} = (\!|[\Gamma]_{u_1=u_2}|\!)$ car les identifiants u_i ne peuvent pas, syntaxiquement, être des variables de récursion. Qui plus est, $[\Gamma]_{u_1=u_2}$ est bien formé si et seulement si $(\!|[\Gamma]_{u_1=u_2}|\!)$ l'est aussi car la compatibilité au sens large considère de la même façon les variables de récursion Z et les variables x_Z. Ces remarques permettent de conclure.

$\mathbf{here}\,(x)\,P'$ **:** On remarque que $[(\!|\Gamma|\!)]_{s\sigma=x} = (\!|[\Gamma]_{s=x}|\!)$, ce qui permet de conclure par hypothèse d'induction.

Autres cas : Ils se prouvent directement grâce à l'hypothèse d'induction et aux propriétés sur les valeurs.

5. Enfin, un raisonnement par induction structurelle sur les systèmes permet d'obtenir le résultat grâce à la propriété sur les processus, en remarquant que $\Gamma = (\!|\Gamma|\!)$ si Γ ne contient aucune variable de récursion.

□

3.3.3 Bisimulations modulo

Les preuves de bisimilarité de deux configurations passent normalement par la création d'une relation qui contient les deux configurations et qui est incluse dans la bisimilarité. Naturellement, les relations choisies dans ce but sont souvent des bisimulations car elles sont toutes immédiatement incluses dans la bisimilarité. Cependant, les bisimulations peuvent s'avérer relativement complexes. Dans le cas qui nous occupe, la traduction implémente par une série d'étapes certaines opérations atomiques. Comme nous l'avons vu à la section 3.3.1, toute bisimulation qui contient à la fois un système et sa traduction doit contenir aussi toutes les étapes intermédiaires.

Or la sémantique du langage suppose que toutes les réductions sauf la communication ne peuvent échouer. Cela signifie intuitivement que si $\Omega \rhd M$ peut effectuer une transition $\Omega \rhd M \xrightarrow{\tau} \Omega \rhd M'$ et que la preuve de cette transition ne contient pas la règle (TE-COMM), les configurations $\Omega \rhd M$ et $\Omega \rhd M'$ sont bisimilaires : les préfixes auxquels peut correspondre cette transition sont effectivement tous inobservables. Puisque ces transitions ne mettent en jeu qu'un seul processus, on parlera de β-réductions, en référence au λ-calcul.

Définition 3.4 (β-réduction)**.** *On dit que la configuration $\Omega \rhd M$ évolue vers la configuration $\Omega \rhd M'$ si la transition $\Omega \rhd M \xrightarrow{\tau} \Omega \rhd M'$ est prouvable et que sa preuve n'utilise pas la règle* (TE-COMM)*. On note alors $\Omega \rhd M \xrightarrow{\tau}_{\beta} \Omega \rhd M'$.*

Afin de réduire de façon drastique le nombre d'éléments dans une bisimulation, les travaux [JR04] et [CHR05] font remarquer que ces étapes ne pouvant échouer[2] peuvent être ignorées. Une preuve de bisimilarité pourra alors se contenter d'exhiber une relation qui est une *bisimulation modulo β-réductions*.

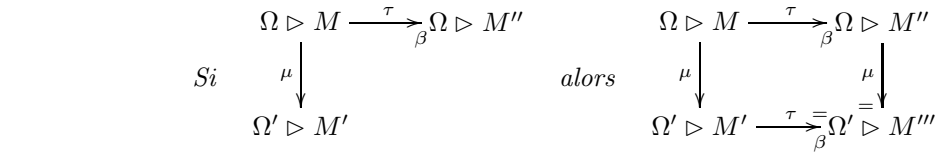
Définition 3.5 (Bisimulation modulo)**.** *Soit $\mathcal{S}$ est une relation typée. On dit que la relation typée $\mathcal{R}$ est une* bisimulation modulo $\mathcal{S}$ *si $\Omega \vDash M \; \mathcal{R} \; N$ implique les deux propriétés suivantes :*

- *si $\Omega \rhd M \xrightarrow{\mu} \Omega$ après $\mu \rhd M'$ alors il doit exister un système N' tel que $\Omega \rhd N \xRightarrow{\hat{\mu}} \Omega$ après $\mu \rhd N'$ et Ω après $\mu \vDash M' \; \mathcal{S} \circ \mathcal{R} \circ \mathcal{S}^{-1} \; N'$;*
- *symétriquement, si $\Omega \rhd N \xrightarrow{\mu} \Omega$ après $\mu \rhd N'$ alors il doit exister un système M' tel que $\Omega \rhd M \xRightarrow{\hat{\mu}} \Omega$ après $\mu \rhd M'$ et Ω après $\mu \vDash M' \; \mathcal{S} \circ \mathcal{R} \circ \mathcal{S}^{-1} \; N'$.*

Ce qui donne à la bisimulation modulo tout son intérêt est bien le fait qu'elle soit incluse dans la bisimilarité. Bien entendu, cette propriété dépend de la relation $\mathcal{S}$ considérée. Dans la suite, on appellera *bisimulation modulo β* la bisimulation modulo $\xrightarrow{\tau}{}^{*}_{\beta}$. La preuve que les bisimulations modulo β sont incluses dans la bisimilarité reposera essentiellement sur la propriété suivante des β-réductions :

[2] Comme on l'a déjà fait remarquer, on suppose en particulier que les migrations ne peuvent échouer. Cette hypothèse devrait être abandonnée dans un cadre où les systèmes peuvent tomber en panne.

Proposition 3.6 (Les β-réductions commutent avec les autres transitions)**.**



Démonstration. Si les deux transitions $\Omega \rhd M \xrightarrow{\mu} \Omega' \rhd M'$ et $\Omega \rhd M \xrightarrow{\tau}_{\beta} \Omega \rhd M''$ sont identiques, le résultat est immédiat. Sinon, par définition des β-réductions, on voit immédiatement que le préfixe réduit par la transition $\Omega \rhd M \xrightarrow{\tau}_{\beta} \Omega \rhd M''$ ne peut être impliqué dans la transition $\Omega \rhd M \xrightarrow{\mu} \Omega' \rhd M'$.

Supposons $\mu = (\Phi)c!V$ et considérons la preuve de la transition $\Omega \rhd M \xrightarrow{(\Phi)c!V} \Omega' \rhd M'$. De l'axiome à la conclusion, cette dérivation doit être de la forme suivante :

- (TE-W) ;
- un paquet de (TE-OUV), (TE-C-PAR) et (TE-C-NEW) ;
- une application de (TE-C-PAR) où le préfixe de la β-réduction est dans la branche non-modifiée de la composition parallèle ; appelons cette occurrence de (TE-C-PAR) *sép* ;
- un dernier paquet de (TE-OUV), (TE-C-PAR) et (TE-C-NEW).

La preuve de la transition $\Omega \rhd M \xrightarrow{\tau}_{\beta} \Omega \rhd M''$, quant à elle, est uniquement constituée d'un axiome suivi de (TE-C-PAR) et (TE-C-NEW). Puisque le préfixe de cette transition est dans la branche non-modifiée de *sép*, cette preuve se récrit directement en une preuve de $\Omega' \rhd M' \xrightarrow{\tau}_{\beta} \Omega' \rhd M'''$ en supprimant les (TE-C-NEW) correspondant aux (TE-OUV) sous *sép* dans la preuve de $\Omega \rhd M \xrightarrow{(\Phi)c!V} \Omega' \rhd M'$. On voit par ailleurs directement que la preuve de $\Omega \rhd M \xrightarrow{(\Phi)c!V} \Omega' \rhd M'$ peut se récrire en changeant la branche non-modifiée pour donner une dérivation $\Omega \rhd M'' \xrightarrow{(\Phi)c!V} \Omega' \rhd M''''$. Qui plus est, $M''' = M''''$ car les mêmes portées de noms sont ouvertes et les mêmes deux préfixes sont réduits, quel que soit l'ordre.

Le même raisonnement peut être tenu pour les autres cas de transition $\Omega \rhd M \xrightarrow{\mu} \Omega' \rhd M'$. □

Comme nous l'avons aussi fait remarquer à la section 3.3.1, la traduction peut contenir des morceaux de processus *morts*, c'est-à-dire incapables de continuer à interagir avec leur environnement. Ainsi, le processus localisé $l[\![\mathsf{stop}]\!]$ est un exemple caractéristique de processus mort. Pour pouvoir supprimer ces morceaux des systèmes, nous utiliserons la bisimilarité forte.

Définition 3.7 (Bisimulation forte)**.** *La relation typée $\mathcal{R}$ est une* bisimulation forte *si $\Omega \vDash M \,\mathcal{R}\, N$ implique les deux propriétés suivantes :*

- *si $\Omega \rhd M \xrightarrow{\mu} \Omega$ $\mathsf{après}$ $\mu \rhd M'$ alors il doit exister un système N' tel que $\Omega \rhd N \xrightarrow{\mu} \Omega$ $\mathsf{après}$ $\mu \rhd N'$ et Ω $\mathsf{après}$ $\mu \vDash M' \,\mathcal{R}\, N'$;*
- *symétriquement, si $\Omega \rhd N \xrightarrow{\mu} \Omega$ $\mathsf{après}$ $\mu \rhd N'$ alors il doit exister un système M' tel que $\Omega \rhd M \xrightarrow{\mu} \Omega$ $\mathsf{après}$ $\mu \rhd M'$ et Ω $\mathsf{après}$ $\mu \vDash M' \,\mathcal{R}\, N'$.*

Définition 3.8 (Bisimilarité forte)**.** *On appelle* bisimilarité forte *la plus grande bisimulation forte. On la note $\sim^t$.*

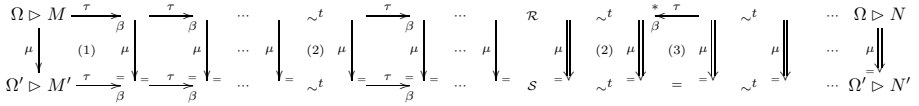
Cette bisimilarité forte se distingue de la bisimilarité normale par le fait que chaque action de l'un des systèmes correspond exactement à une action de l'autre. Cela signifie en particulier que les systèmes totalement inactifs $l[\![\mathsf{stop}]\!]$

et **0** sont naturellement fortement bisimilaires. Par ailleurs, le lemme 2.38 permet de conclure immédiatement que la congruence structurelle est incluse dans la bisimilarité forte.

En combinant alors les β-réductions et la bisimilarité forte, on obtient la bisimulation modulo $\xrightarrow{\tau}_{\beta}^{*}\sim^{t}$. Prouvons alors que toute bisimulation modulo β et bisimilarité forte est effectivement incluse dans la bisimilarité.

Proposition 3.9 (Une bisimulation modulo β et bisimilarité forte est incluse dans la bisimilarité)**.** *Si $\mathcal{R}$ est une bisimulation modulo $\xrightarrow{\tau}_{\beta}^{*}\sim^{t}$, alors $\mathcal{R} \subseteq \approx^{t}$.*

Démonstration. Prouvons donc que $(\xrightarrow{\tau}_{\beta}^{*}\sim^{t})^{*} \circ \mathcal{R} \circ ((\xrightarrow{\tau}_{\beta}^{*}\sim^{t})^{*})^{-1}$ est une bisimulation. Le fait que cette relation contienne $\mathcal{R}$ permettra alors de conclure. En notant $\mathcal{S}$ la relation $(\xrightarrow{\tau}_{\beta}^{*}\sim^{t})^{*} \circ \mathcal{R} \circ ((\xrightarrow{\tau}_{\beta}^{*}\sim^{t})^{*})^{-1}$, la preuve se construit de la façon suivante :



où

(1) est prouvé par la proposition 3.6 car les commutations de transitions correspondent directement à des commutations d'actions ;

(2) est prouvé par la définition même de bisimulation forte ;

(3) découle directement du fait que les transitions faibles $\stackrel{\mu}{\Longrightarrow}^{=}$ peuvent contenir un nombre arbitraire de transitions $\xrightarrow{\tau}$, donc en particulier $\xrightarrow{\tau}_{\beta}$.

Le diagramme utilise de plus le fait que $\stackrel{\hat{\mu}}{\Longrightarrow}^{=}$ et $\stackrel{\mu}{\Longrightarrow}^{=}$ coïncident. □

Cette propriété permet de vérifier rapidement une remarque initiale : les β-réductions sont incluses dans la bisimilarité.

Corollaire 3.10. *Soit une transition $\Omega \rhd M \xrightarrow{\tau}_{\beta} \Omega \rhd M'$. Alors $\Omega \vDash M \approx^{t} M'$.*

Démonstration. Ce résultat est immédiat en constatant que la simple relation typée $\{(\Omega, M, M')\}^{=}$ est une bisimulation modulo β-réductions, en utilisant des commutations entre les β-réductions et toute autre transition. □

3.3.4 Traduction annotée

Suivant les difficultés soulevées à la section 3.3.1, il nous faut encore donner une traduction bien plus précise que l'implémentation elle-même, dans laquelle la localité refuge associée à certains processus récursifs a déjà été créée. Par rapport à la traduction initiale, on rajoute une annotation indiquant pour chaque processus récursif du système si sa localité existe déjà ou non. Afin de pouvoir identifier précisément chaque occurrence d'un processus récursif, on définit une notion de *position* correspondant à cette occurrence, exactement comme l'on avait défini le sous-arbre situé à une position donnée dans un type (définition 2.13).

Définition 3.11 (Position dans un processus, un système)**.** *Si p est un mot sur l'alphabet $\{0, 1, 2\}$, le processus situé à la* position *p dans P, noté $P|_{p}$, est défini inductivement par les équations suivantes :*

– $P|_{\varepsilon} = P$;

- $(u\,!\,\langle V\rangle\,P)|_{0p} = P|_p$;
- $(u\,?\,(X:\mathsf{T})\,P)|_{0p} = P|_p$;
- $(\mathit{goto}\,u.\;P)|_{0p} = P|_p$;
- $(\mathit{if}\,u_1 = u_2\ \mathit{then}\,P_1\ \mathit{else}\,P_2)|_{1p} = P_1|_p$;
- $(\mathit{if}\,u_1 = u_2\ \mathit{then}\,P_1\ \mathit{else}\,P_2)|_{2p} = P_2|_p$;
- $(\mathit{newchan}\,c : \mathsf{C}\ \mathit{in}\,P)|_{0p} = P|_p$;
- $(\mathit{newloc}\,k, (c_1,\ldots,c_n) : \mathsf{L}\ \mathit{with}\,P_k\ \mathit{in}\,P)|_{1p} = P_k|_p$;
- $(\mathit{newloc}\,k, (c_1,\ldots,c_n) : \mathsf{L}\ \mathit{with}\,P_k\ \mathit{in}\,P)|_{2p} = P|_p$;
- $(P_1 \mid P_2)|_{1p} = P_1|_p$;
- $(P_1 \mid P_2)|_{2p} = P_2|_p$;
- $(\mathit{here}\,(x)\,P)|_{0p} = P|_p$;
- $(*P)|_{0p} = P|_p$;
- $((\mathit{rec}\,Z : \mathsf{R}\,(X)\,.\;P)\,\langle V\rangle)|_{0p} = P|_p$.

Il est indéfini dans les autres cas, notamment $Z\,\langle V\rangle|_p$ *et* $\mathsf{stop}|_p$ *lorsque* p *n'est pas le mot vide* ε.

De la même façon, on définit le système ou processus situé à la position p *dans* M*, noté* $M|_p$*, par les équations suivantes :*

- $M|_\varepsilon = M$;
- $(l[\![P]\!])|_{0p} = P|_p$;
- $(M_1 \mid M_2)|_{1p} = M_1|_p$;
- $(M_1 \mid M_2)|_{2p} = M_2|_p$;
- $((\mathit{new}\,a : \mathsf{E})\,M)|_{0p} = M|_p$;

$M|_p$ *est indéfini dans les autres cas.*

Comme les exemples de la section 3.3.1 l'avaient mis en évidence, deux occurrences d'un processus récursif pourront découler d'un même processus récursif initial et donc se voir associer la même localité refuge dans la traduction. La notion de position permet de prendre en compte ce phénomène en paramétrant la traduction par une annotation indiquant l'ensemble des positions de processus récursifs qui ont la même localité refuge associée. Cette annotation contiendra donc des ensembles de positions : toutes les positions d'un tel ensemble correspondront à un même processus récursif initial.

On dira que $\mathcal{P}$ est une sous-partition annotée de l'ensemble des positions du système M si $\mathcal{P}$ est une liste de $E_i : n_i$ avec les conditions suivantes.

- E_i est un ensemble de positions du système M ; n_i est un entier naturel.
- $E_i \cap E_j = \emptyset$ si $i \neq j$.
- Pour tout $E : n \in \mathcal{P}$, et pour toute position $p \in E$, $M|_p$ est défini et prend la forme
  - d'un appel récursif $Z\,\langle V\rangle$,
  - ou d'un processus récursif $(\mathsf{rec}\,Z : \mathsf{R}\,(X)\,.\;P)\,\langle V\rangle$ avec $\mathsf{fv}(P) \subset \mathsf{fv}(X)$ et tel que tous les noms a de $\mathsf{fn}(P)$ soient libres dans M ou liés par une construction $(\mathsf{new}\,a : \mathsf{E})$ (c'est-à-dire pas par $\mathsf{newchan}\,a$ ou $\mathsf{newloc}\,a$).
- Si $p \in E$, $E : n \in \mathcal{P}$ avec $M|_p = (\mathsf{rec}\,Z : \mathsf{R}\,(X)\,.\;P)\,\langle V\rangle$ alors toutes les positions des appels récursifs sont aussi dans E, c'est-à-dire pour tout p' tel que $M|_{pp'} = Z\,\langle V'\rangle$, $pp' \in E$. Réciproquement, si $pp' \in E \in \mathcal{P}$ est telle que $M|_p = (\mathsf{rec}\,Z : \mathsf{R}\,(X)\,.\;P)\,\langle V\rangle$ et $M|_{pp'} = Z\,\langle V'\rangle$ alors $p \in E$.
- Si $p \in E$, $E : n \in \mathcal{P}$ avec $M|_p = (\mathsf{rec}\,Z : \mathsf{R}\,(X)\,.\;P)\,\langle V\rangle$ alors toutes les variables de récursion libres dans P sont à une position de $\mathcal{P}$, c'est-à-dire que, pour tout p' tel que $M|_{pp'} = Z'\,\langle V'\rangle$, il existe $E' : n'$ avec $pp' \in E'$ et $E' : n' \in \mathcal{P}$.

Pour une position p, on écrira $p \in \mathcal{P}$ s'il existe un $E_i : n_i$ dans $\mathcal{P}$ avec $p \in E_i$.

Afin de simplifier notablement la preuve de l'équivalence entre un système et sa traduction annotée, la traduction ne sera définie que sur les systèmes de la forme $\mathfrak{C}_M[M]$ où $\mathfrak{C}_M$ est de la forme $(\mathsf{new}\, a_1 : \mathsf{E}_1) \cdots (\mathsf{new}\, a_n : \mathsf{E}_n)[\cdot]$ et M ne contient aucun new. Bien entendu, tout système est structurellement congru à un système de cette forme donc ce choix n'aura pour effet que de simplifier la preuve. Pour définir la nouvelle traduction du système $\mathfrak{C}_M[M]$, commençons par définir la traduction du système M, qui ne contient donc aucun new, annotée par la sous-partition $\mathcal{P}$. Cette traduction est aussi annotée par la position courante dans M, de sorte qu'il soit possible de savoir si le processus récursif à traduire est à une position apparaissant dans la sous-partition $\mathcal{P}$. La traduction annotée de M sera notée $(\!|M|\!)^{\varepsilon}_{\mathcal{P}}$ et définie structurellement comme suit.

- Si $p0 \in \mathcal{P}$, on note E l'ensemble tel que $E : n$ apparaisse dans $\mathcal{P}$ et $p0 \in E$ et on définit :

$$(\!|l[\![(\mathsf{rec}\ Z : \mathsf{R}\,(X)\,.\ P)\,\langle V\rangle]\!|)^{p}_{\mathcal{P}} = ref_{Z_E}[\![inst_{Z_E}\,!\,\langle (l, V)\rangle]\!]$$

  La traduction du système $l[\![(\mathsf{rec}\ Z : \mathsf{R}\,(X)\,.\ P)\,\langle V\rangle]\!]$ n'est pas définie dans le cas où $p0 \notin \mathcal{P}$.
- Si le cas précédent ne s'applique pas car $(\mathsf{rec}\ Z : \mathsf{R}\,(X)\,.\ P)\,\langle V\rangle$ apparaît au sein d'un processus (c'est-à-dire sous un préfixe) mais que $p \in \mathcal{P}$, on note E l'ensemble tel que $E : n$ apparaisse dans $\mathcal{P}$ et $p \in E$ et on définit :

$$(\!|(\mathsf{rec}\ Z : \mathsf{R}\,(X)\,.\ P)\,\langle V\rangle|\!)^{p}_{\mathcal{P}} = \mathsf{here}\,(x_{Z_E})\,\mathsf{goto}\ ref_{Z_E}.\ inst_{Z_E}\,!\,\langle x_{Z_E}, V\rangle$$

- Si $p \notin \mathcal{P}$ mais que $(\mathsf{rec}\ Z : \mathsf{R}\,(X)\,.\ P)\,\langle V\rangle$ apparaît sous un préfixe, la traduction est calquée sur la version non annotée fournie dans la section 3.2. En particulier, le déclencheur $(\!|Z\,\langle V\rangle|\!)$ est conservé intact. Dans ce cas, la traduction prend la forme :

$$\begin{array}{l}(\!|(\mathsf{rec}\ Z : \mathsf{R}\,(X)\,.\ P)\,\langle V\rangle|\!)^{p}_{\mathcal{P}} = \\ \quad \mathsf{newloc}\ ref_Z, inst_Z : \sum x : \text{LOC. RW}\langle \mathsf{R}\rangle_{@x}\, x\ \mathsf{with} \\ \qquad *inst_Z\,?\,((x_Z, X) : \mathsf{R})\ \mathsf{goto}\ x_Z.\ (\!|P|\!)^{p0}_{\mathcal{P}}\ \mathsf{in} (\!|Z\,\langle V\rangle|\!)\end{array}$$

- Si $p \in \mathcal{P}$, on note E l'ensemble tel que $E : n$ apparaisse dans $\mathcal{P}$ et $p \in E$ et on définit :

$$(\!|Z\,\langle V\rangle|\!)^{p}_{\mathcal{P}} = \mathsf{here}\,(x_{Z_E})\,\mathsf{goto}\ ref_{Z_E}.\ inst_{Z_E}\,!\,\langle x_{Z_E}, V\rangle$$

- Si p n'est pas dans $\mathcal{P}$, on définit :

$$(\!|Z\,\langle V\rangle|\!)^{p}_{\mathcal{P}} = \mathsf{here}\,(x_Z)\,\mathsf{goto}\ ref_Z.\ inst_Z\,!\,\langle x_Z, V\rangle$$

- De la même façon que la traduction initiale $(\!|P|\!)$, cette traduction annotée passe sur les autres constructions sans les modifier, en mettant simplement à jour l'information de position courante dans le processus en cours de traduction.

$$(\!|u\,!\,\langle V\rangle\ P|\!)^{p}_{\mathcal{P}} = u\,!\,\langle V\rangle\ (\!|P|\!)^{p0}_{\mathcal{P}}$$
$$(\!|u\,?\,(X : \mathsf{T})\ P|\!)^{p}_{\mathcal{P}} = u\,?\,(X : \mathsf{T})\ (\!|P|\!)^{p0}_{\mathcal{P}}$$

- Cette traduction traite comme on l'attend les systèmes de la forme $l[\![P]\!]$ quand P n'est pas un processus récursif :

$$(\!|l[\![P]\!]|\!)^{p}_{\mathcal{P}} = l[\![(\!|P|\!)^{p0}_{\mathcal{P}}]\!]$$

Elle procède aussi comme on l'attend sur les systèmes de la forme $M_1 \mid M_2$:

$$(\!|M_1 \mid M_2|\!)^p_{\mathcal{P}} = (\!|M_1|\!)^{p1}_{\mathcal{P}} \mid (\!|M_2|\!)^{p2}_{\mathcal{P}}$$

La traduction du système complet $\mathfrak{C}_M[M]$ doit, quant à elle, rajouter les localités refuges qui ont déjà été créées par les processus. Si $\mathcal{P}$ est la liste $E_1 : n_1, E_2 : n_2, \ldots$ et que $p_i \in E_i$, on définit alors le contexte $\mathfrak{C}_T$ de la façon suivante :

$$\begin{array}{l}
\mathfrak{C}_T = \\
\quad (\mathsf{new}\, ref_{Z_{E_1}} : \textsc{loc})\,(\mathsf{new}\, inst_{Z_{E_1}} : \textsc{rw}\langle \mathsf{R}_{E_1}\rangle_{@} x) \\
\quad (\mathsf{new}\, ref_{Z_{E_2}} : \textsc{loc})\,(\mathsf{new}\, inst_{Z_{E_2}} : \textsc{rw}\langle \mathsf{R}_{E_2}\rangle_{@} x) \\
\quad\quad \vdots \\
\quad\quad\quad ref_{Z_{E_1}}[\![inst_{Z_{E_1}}\,?\,\big((x_{Z_{E_1}}, X) : \mathsf{R}_{E_1}\big)\,\mathsf{goto}\, x_{Z_{E_1}}\,.\,(\!|(M)|_{p_1 0}|\!)^{p_1 0}_{\mathcal{P}}]\!] \\
\quad\quad\quad\quad \vdots \times (n_1 + 1) \\
\quad\quad | \quad ref_{Z_{E_1}}[\![inst_{Z_{E_1}}\,?\,\big((x_{Z_{E_1}}, X) : \mathsf{R}_{E_1}\big)\,\mathsf{goto}\, x_{Z_{E_1}}\,.\,(\!|(M)|_{p_1 0}|\!)^{p_1 0}_{\mathcal{P}}]\!] \\
\quad\quad | \quad ref_{Z_{E_1}}[\![{*}inst_{Z_{E_1}}\,?\,\big((x_{Z_{E_1}}, X) : \mathsf{R}_{E_1}\big)\,\mathsf{goto}\, x_{Z_{E_1}}\,.\,(\!|(M)|_{p_1 0}|\!)^{p_1 0}_{\mathcal{P}}]\!] \\
\quad\quad | \quad ref_{Z_{E_2}}[\![inst_{Z_{E_2}}\,?\,\big((x_{Z_{E_2}}, X) : \mathsf{R}_{E_2}\big)\,\mathsf{goto}\, x_{Z_{E_2}}\,.\,(\!|(M)|_{p_2 0}|\!)^{p_2 0}_{\mathcal{P}}]\!] \\
\quad\quad\quad\quad \vdots \times (n_2 + 1) \\
\quad\quad | \quad ref_{Z_{E_2}}[\![inst_{Z_{E_2}}\,?\,\big((x_{Z_{E_2}}, X) : \mathsf{R}_{E_2}\big)\,\mathsf{goto}\, x_{Z_{E_2}}\,.\,(\!|(M)|_{p_2 0}|\!)^{p_2 0}_{\mathcal{P}}]\!] \\
\quad\quad | \quad ref_{Z_{E_2}}[\![{*}inst_{Z_{E_2}}\,?\,\big((x_{Z_{E_2}}, X) : \mathsf{R}_{E_2}\big)\,\mathsf{goto}\, x_{Z_{E_2}}\,.\,(\!|(M)|_{p_2 0}|\!)^{p_2 0}_{\mathcal{P}}]\!] \\
\quad\quad\quad\quad \vdots \\
\quad\quad | \quad [\cdot]
\end{array}$$

où R_{E_i} est le type associé à la variable de récursion Z_{E_i} apparaissant à la position p_i. On en déduit la traduction suivante pour les systèmes complets :

$${}_{sys}(\!|\mathfrak{C}_M[M]|\!)_{\mathcal{P}} = \mathfrak{C}_M[\mathfrak{C}_T[(\!|M|\!)^{\varepsilon}_{\mathcal{P}}]]$$

On remarque que le contexte $\mathfrak{C}_T$ dépend du choix de p_i parmi toutes les positions de l'ensemble E_i. On rajoutera donc explicitement une condition sur la sous-partition annotée $\mathcal{P}$: le contexte $\mathfrak{C}_T$ associé au système $\mathfrak{C}_M[M]$ et à la sous-partition $\mathcal{P}$ doit être identique quel que soit le p_i choisi comme représentant de l'ensemble E_i de $\mathcal{P}$. Le sens de cette condition est d'assurer que toutes les positions dans un E de $\mathcal{P}$ peuvent effectivement résulter d'un même processus récursif initial. Cette condition sera par conséquent automatiquement vérifiée dans les cas qui découlent de la traduction normale $(\!|\mathfrak{C}_M[M]|\!)$, puisque les localités-refuges sont créées à la volée.

3.3.5 Équivalence proprement dite

Malgré tous les outils développés dans les sections qui précèdent, quelques notions préalables doivent être mises en place pour prouver le résultat d'équivalence. Le premier des problèmes restant à résoudre vient du fait que la sous-partition qui paramétrise la traduction doit être mise à jour lorsqu'une transition est effectuée. Par exemple au cours de la transition

$$l[\![a\,?\,()\,b\,?\,()\,(\mathsf{rec}\; Z : \mathsf{R}\,(X)\,.\,P)\,\langle V\rangle]\!] \xrightarrow{a?} l[\![b\,?\,()\,(\mathsf{rec}\; Z : \mathsf{R}\,(X)\,.\,P)\,\langle V\rangle]\!]$$

la position du processus récursif passe de 000 à 00. L'évolution est bien entendu plus complexe lorsqu'ont lieu des réplications, des récursions, etc. Afin de suivre l'évolution des positions, on définit le résidu d'une position de processus qui sera l'ensemble des positions auquel le processus se retrouve après cette transition. Le résidu tient compte du choix qui exclut les systèmes contenant des new : dans le cas où un préfixe newchan ou newloc est réduit, l'impact sur les positions que devraient avoir les new engendrés est totalement oublié. En effet, les systèmes après réduction seront renormalisés sous la forme $\mathfrak{C}_{M'}[M']$ où les new n'influent pas sur les positions dans M'.

Définition 3.12 (Résidu d'une position de processus après une transition)**.** *On appelle* résidu *d'une position de processus p d'un système M après une transition* $\Omega \rhd M \xrightarrow{\mu} \Omega' \rhd M'$ *l'ensemble de positions* $\mathsf{Rés}(p, \Omega \rhd M \xrightarrow{\mu} \Omega' \rhd M')$*, où la définition de la fonction* Rés *dépend de la preuve de la transition et est fournie par les équations suivantes. Seule la conclusion de la preuve de la transition est mentionnée dans ces équations.*

- $\mathsf{Rés}(\varepsilon, \Omega \rhd M \xrightarrow{\mu} \Omega' \rhd M') = \{\varepsilon\}$;
- $\mathsf{Rés}(00p, \Omega \rhd l[\![a\,!\,\langle V\rangle\, P]\!] \xrightarrow{a!V} \Omega' \rhd l[\![P]\!]) = \{0p\}$;
- $\mathsf{Rés}(00p, \Omega \rhd l[\![a\,?\,(X:\mathsf{T})\, P]\!] \xrightarrow{a?V} \Omega \rhd l[\![P\{^V/_X\}]\!]) = \{0p\}$;
- $\mathsf{Rés}(1p, \Omega \rhd M \mid N \xrightarrow{\tau} \Omega \rhd (\mathsf{new}\,\Phi)\, M' \mid N') = \{1p' \mid p' \in \mathsf{Rés}(p, \Omega_M \rhd M \xrightarrow{\mu} \Omega'_M \rhd M')\}$ *que* μ *soit une émission ou une réception ;*
- $\mathsf{Rés}(2p, \Omega \rhd M \mid N \xrightarrow{\tau} \Omega \rhd (\mathsf{new}\,\Phi)\, M' \mid N') = \{2p' \mid p' \in \mathsf{Rés}(p, \Omega_N \rhd N \xrightarrow{\mu} \Omega'_N \rhd N')\}$ *que* μ *soit une émission ou une réception ;*
- $\mathsf{Rés}(00p, \Omega \rhd l[\![\mathsf{goto}\,k.\ P]\!] \xrightarrow{\tau} \Omega \rhd k[\![P]\!]) = \{0p\}$;
- $\mathsf{Rés}(01p, \Omega \rhd l[\![\mathsf{if}\,a = a\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2]\!] \xrightarrow{\tau} \Omega \rhd l[\![P_1]\!]) = \{0p\}$;
- $\mathsf{Rés}(02p, \Omega \rhd l[\![\mathsf{if}\,a_1 = a_2\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2]\!] \xrightarrow{\tau} \Omega \rhd l[\![P_2]\!]) = \{0p\}$ *quand* $a_1 \neq a_2$;
- $\mathsf{Rés}(00p, \Omega \rhd l[\![\mathsf{newchan}\,c : \mathsf{C}\ \mathsf{in}\ P]\!] \xrightarrow{\tau} \Omega \rhd (\mathsf{new}\,c : \mathsf{C}_{@l})\, l[\![P]\!]) = \{0p\}$;
- $\mathsf{Rés}(01p, \Omega \rhd l[\![\mathsf{newloc}\,k, (\vec{c}) : \mathsf{L}\ \mathsf{with}\ P_k\ \mathsf{in}\ P]\!] \xrightarrow{\tau} \Omega \rhd (\mathsf{new}\langle(k, (\vec{c})) : \mathsf{L}\rangle)\, k[\![P_k]\!] \mid l[\![P]\!]) = \{10p\}$;
- $\mathsf{Rés}(02p, \Omega \rhd l[\![\mathsf{newloc}\,k, (\vec{c}) : \mathsf{L}\ \mathsf{with}\ P_k\ \mathsf{in}\ P]\!] \xrightarrow{\tau} \Omega \rhd (\mathsf{new}\langle(k, (\vec{c})) : \mathsf{L}\rangle)\, k[\![P_k]\!] \mid l[\![P]\!]) = \{20p\}$;
- $\mathsf{Rés}(01p, \Omega \rhd l[\![P_1 \mid P_2]\!] \xrightarrow{\tau} \Omega \rhd l[\![P_1]\!] \mid l[\![P_2]\!]) = \{10p\}$;
- $\mathsf{Rés}(02p, \Omega \rhd l[\![P_1 \mid P_2]\!] \xrightarrow{\tau} \Omega \rhd l[\![P_1]\!] \mid l[\![P_2]\!]) = \{20p\}$;
- $\mathsf{Rés}(00p, \Omega \rhd l[\![\mathsf{here}\,(x)\, P]\!] \xrightarrow{\tau} \Omega \rhd l[\![P\{^l/_x\}]\!]) = \{0p\}$;
- $\mathsf{Rés}(0, \Omega \rhd l[\![*P]\!] \xrightarrow{\tau} \Omega \rhd l[\![P]\!] \mid l[\![*P]\!]) = \{20\}$;
- $\mathsf{Rés}(00p, \Omega \rhd l[\![*P]\!] \xrightarrow{\tau} \Omega \rhd l[\![P]\!] \mid l[\![*P]\!]) = \{10p, 200p\}$;
- $\mathsf{Rés}(0, \Omega \rhd l[\![(\mathsf{rec}\ Z : \mathsf{R}\,(X)\,.\ P)\,\langle V\rangle]\!] \xrightarrow{\tau} \Omega \rhd l[\![P\{^V/_X\}[^{\mathsf{rec}\ Z:\mathsf{R}(X).\ P}/_Z]]\!]) = \{0p \mid P|_p = Z\,\langle V'\rangle\}$;
- $\mathsf{Rés}(00p, \Omega \rhd l[\![(\mathsf{rec}\ Z : \mathsf{R}\,(X)\,.\ P)\,\langle V\rangle]\!] \xrightarrow{\tau} \Omega \rhd l[\![P\{^V/_X\}[^{\mathsf{rec}\ Z:\mathsf{R}(X).\ P}/_Z]]\!]) = \{0p, 0p'0p \mid P|_{p'} = Z\,\langle V'\rangle\}$ *si* $P|_p$ *est défini ;*
- $\mathsf{Rés}(1p, \Omega \rhd M \mid N \xrightarrow{\mu} \Omega' \rhd M' \mid N) = \{1p' \mid p' \in \mathsf{Rés}(p, \Omega \rhd M \xrightarrow{\mu} \Omega' \rhd M')\}$;
- $\mathsf{Rés}(2p, \Omega \rhd M \mid N \xrightarrow{\mu} \Omega' \rhd M \mid N') = \{2p' \mid p' \in \mathsf{Rés}(p, \Omega \rhd N \xrightarrow{\mu} \Omega' \rhd N')\}$;
- $\mathsf{Rés}(1p, \Omega \rhd M \mid N \xrightarrow{\mu} \Omega' \rhd M \mid N') = \{1p\}$;
- $\mathsf{Rés}(2p, \Omega \rhd M \mid N \xrightarrow{\mu} \Omega' \rhd M' \mid N) = \{2p\}$;
- $\mathsf{Rés}(0p, \Omega \rhd (\mathsf{new}\,a : \mathsf{E})\, M \xrightarrow{\mu} \Omega' \rhd (\mathsf{new}\,a : \mathsf{E})\, M') = \{0p' \mid p' \in \mathsf{Rés}(p, \Omega \rhd M \xrightarrow{\mu} \Omega' \rhd M')\}$;
- $\mathsf{Rés}(0p, \Omega \rhd (\mathsf{new}\,b : \mathsf{E})\, M \xrightarrow{(b:\mathsf{E};\Phi)a!V} \Omega' \rhd M') = \mathsf{Rés}(p, \Omega \rhd M \xrightarrow{(\Phi)a!V} \Omega' \rhd M')$;
- $\mathsf{Rés}(p, \Omega \rhd M \xrightarrow{(b:\mathsf{E};\Phi)a?V} \Omega' \rhd M') = \mathsf{Rés}(p, \Omega; b : \mathsf{E} \rhd M \xrightarrow{(\Phi)a?V} \Omega' \rhd M')$.

Dans tous les autres cas, Rés *prend la valeur* $\emptyset$.

On étend la fonction Rés aux ensembles de positions et aux sous-partitions de la façon suivante :

$$\mathsf{Rés}(E, \Omega \rhd M \xrightarrow{\mu} \Omega' \rhd M') = \bigcup_{p \in E} \mathsf{Rés}(p, \Omega \rhd M \xrightarrow{\mu} \Omega' \rhd M')$$

$$\mathsf{Rés}((\mathcal{P}, E : n), \Omega \rhd M \xrightarrow{\mu} \Omega' \rhd M') = \mathsf{Rés}(\mathcal{P}, \Omega \rhd M \xrightarrow{\mu} \Omega' \rhd M') \quad \text{si } \mathsf{Rés}(E, \Omega \rhd M \xrightarrow{\mu} \Omega' \rhd M') = \emptyset$$

$$\mathsf{Rés}((\mathcal{P}, E : n), \Omega \rhd M \xrightarrow{\mu} \Omega' \rhd M') = \mathsf{Rés}(\mathcal{P}, \Omega \rhd M \xrightarrow{\mu} \Omega' \rhd M'), \mathsf{Rés}(E, \Omega \rhd M \xrightarrow{\mu} \Omega' \rhd M') : n \quad \text{sinon}$$

Le dernier problème non encore traité est celui des processus répliqués dont le processus récursif associé a disparu. Le lemme suivant permettra de gérer la situation.

Lemme 3.13 (Nettoyage des systèmes)**.** *Soit le système*

$$M = (\mathsf{new}\, l : \text{LOC})\,(\mathsf{new}\, c : \text{RW}\langle \mathsf{T} \rangle_{@} l) \\ (l[\![c\,?\,(X : \mathsf{T})\,P]\!] \mid \cdots l[\![c\,?\,(X : \mathsf{T})\,P]\!] \mid l[\![*c\,?\,(X : \mathsf{T})\,P]\!] \mid N)$$

tel que $l, c \notin \mathsf{fn}(N)$ *et que* $\Omega \rhd M$ *soit bien formée. On a alors :*

$$\Omega \vDash M \sim^t N \mid (\mathsf{new}\, l : \text{LOC})\, l[\![* \mathsf{stop}]\!]$$

Démonstration. Considérons la relation $\mathcal{R}$ définie par

$$\Omega \vDash M \;\mathcal{R}\; N \mid (\mathsf{new}\, l : \text{LOC})\, l[\![\mathsf{stop}]\!] \mid \cdots \mid l[\![\mathsf{stop}]\!] \mid l[\![* \mathsf{stop}]\!]$$

telle que $l, c \notin \mathsf{fn}(N)$ et que $\Omega \rhd M$ soit bien formée avec

$$M = (\mathsf{new}\, l : \text{LOC})\,(\mathsf{new}\, c : \text{RW}\langle \mathsf{T} \rangle_{@} l) \\ (l[\![c\,?\,(X : \mathsf{T})\,P]\!] \mid \cdots \mid l[\![c\,?\,(X : \mathsf{T})\,P]\!] \mid l[\![*c\,?\,(X : \mathsf{T})\,P]\!] \mid N)$$

On voit aisément que cette relation est une bisimulation en constatant que les portées des noms l et c ne peuvent jamais être ouvertes, donc toute action effectuée par un système de la forme M donne un système qui possède toujours cette forme. Par ailleurs, on sait que cette action ne peut être effectuée que deux façons : par N seul, auquel cas elle est aisément simulée ; ou bien par la réplication de $l[\![*c\,?\,(X : \mathsf{T})\,P]\!]$, auquel cas $l[\![* \mathsf{stop}]\!]$ peut effectuer une transition similaire. □

Nous disposons enfin de tous les outils pour prouver le lemme essentiel de la question de l'équivalence.

Lemme 3.14 (La traduction annotée est une bisimulation)**.** *Si la configuration* $\Omega \rhd \mathfrak{C}_M[M]$ *est bien formée et que la traduction* ${}_{sys}(\!|\mathfrak{C}_M[M]|\!)_{\mathcal{P}}$ *est définie,*

$$\Omega \vDash \mathfrak{C}_M[M] \approx^t {}_{sys}(\!|\mathfrak{C}_M[M]|\!)_{\mathcal{P}} \mid (\mathsf{new}\, l : \text{LOC})\, l[\![* \mathsf{stop}]\!]$$

Démonstration. Remarquons tout d'abord que la configuration $\Omega \rhd \mathfrak{C}_M[M]$ est bien formée si et seulement si $\Omega \rhd {}_{sys}(\!|\mathfrak{C}_M[M]|\!)_{\mathcal{P}}$ l'est également, par un raisonnement très similaire à la proposition 3.3. Cela autorise donc à envisager que ces deux configurations soient bisimilaires.

Prouvons donc que la relation $\mathcal{R}$ contenant

$$\Omega \vDash \mathfrak{C}_M[M] \;\mathcal{R}\; {}_{sys}(\!|\mathfrak{C}_M[M]|\!)_{\mathcal{P}} \mid (\mathsf{new}\, l : \textsc{loc})\, l[\![* \,\mathsf{stop}]\!]$$

quand $\Omega \rhd \mathfrak{C}_M[M]$ est bien formée et ${}_{sys}(\!|\mathfrak{C}_M[M]|\!)_{\mathcal{P}}$ est définie, est une bisimulation modulo β-réduction et bisimilarité forte.

Supposons donc tout d'abord

$$\Omega \rhd \mathfrak{C}_M[M] \xrightarrow{\mu} \Omega' \rhd \mathfrak{C}'_M[M']$$

et prouvons

$$\Omega \rhd {}_{sys}(\!|\mathfrak{C}_M[M]|\!)_{\mathcal{P}} \xrightarrow{\mu} \xrightarrow{\tau}{}^*_{\beta} \sim^t \Omega' \rhd {}_{sys}(\!|\mathfrak{C}'_M[M']|\!)_{\mathcal{P}'}$$

ou

$$\Omega \rhd {}_{sys}(\!|\mathfrak{C}_M[M]|\!)_{\mathcal{P}} \xrightarrow{\mu} \xrightarrow{\tau}{}^*_{\beta} \sim^t \Omega' \rhd {}_{sys}(\!|\mathfrak{C}'_M[M']|\!)_{\mathcal{P}'} \mid (\mathsf{new}\, l : \textsc{loc})\, l[\![* \,\mathsf{stop}]\!]$$

Puisque la relation recherchée est une bisimulation modulo bisimilarité forte et que les systèmes

$$(\mathsf{new}\, l : \textsc{loc})\, l[\![* \,\mathsf{stop}]\!] \mid (\mathsf{new}\, l : \textsc{loc})\, l[\![* \,\mathsf{stop}]\!]$$

et

$$(\mathsf{new}\, l : \textsc{loc})\, l[\![* \,\mathsf{stop}]\!]$$

sont naturellement fortement bisimilaires, cette action suffira à conclure.

L'action du système $\Omega \rhd \mathfrak{C}_M[M]$ repose nécessairement sur une action effectuée par $\Omega \rhd M$. On peut distinguer deux cas : la preuve de cette action peut ne contenir qu'un axiome ou être une communication. Commençons en distinguant les différents axiomes possibles :

(te-r) La règle prouvant la transition doit être de la forme

$$\Omega \rhd l[\![c\,?\,(X : \mathsf{T})\, P]\!] \xrightarrow{c?V} \Omega \rhd l[\![P\{^V/_X\}]\!]$$

quand

$$\Omega \rhd \mathfrak{C}_M[M] \xrightarrow{(\Phi)c?V} \Omega; \Phi \rhd \mathfrak{C}_M[M']$$

On note $M = \mathfrak{C}_?[l[\![c\,?\,(X : \mathsf{T})\, P]\!]]$.

Si $l[\![c\,?\,(X : \mathsf{T})\, P]\!]$ est situé à la position p dans M, la traduction doit vérifier :

$$(\!|l[\![c\,?\,(X : \mathsf{T})\, P]\!]|\!)^p_{\mathcal{P}} = l[\![c\,?\,(X : \mathsf{T})\, (\!|P|\!)^{p00}_{\mathcal{P}}]\!]$$

c'est-à-dire que la traduction du système complet doit avoir la forme suivante :

$$\begin{array}{rcl} {}_{sys}(\!|\mathfrak{C}_M[\mathfrak{C}_?[l[\![c\,?\,(X : \mathsf{T})\, P]\!]]]|\!)^{\varepsilon}_{\mathcal{P}} & = & \mathfrak{C}_M[\mathfrak{C}_T[(\!|\mathfrak{C}_?[l[\![c\,?\,(X : \mathsf{T})\, P]\!]]]|\!)^{\varepsilon}_{\mathcal{P}}] \\ & = & \mathfrak{C}_M[\mathfrak{C}_T[\mathfrak{C}'_?[l[\![c\,?\,(X : \mathsf{T})\, (\!|P|\!)^{p00}_{\mathcal{P}}]\!]]] \end{array}$$

On en déduit la transition suivante :

$$\Omega \rhd l[\![c\,?\,(X : \mathsf{T})\, (\!|P|\!)^{p00}_{\mathcal{P}}]\!] \xrightarrow{c?V} \Omega \rhd l[\![(\!|P|\!)^{p00}_{\mathcal{P}}\{^V/_X\}]\!]$$

On voit aisément que la traduction commute avec la substitution de variables car les processus récursifs déjà traduits sous forme de processus

répliqués, c'est-à-dire dont la position est dans $\mathcal{P}$, sont clos. Par conséquent cette transition aboutit à la configuration $\Omega \rhd l[\![(\!|P\{^V/_X\}|\!)^{p00}_{\mathcal{P}}]\!]$. Les résidus de toutes les positions de la forme $p00p_0$ après cette transition sont naturellement $\{p0p_0\}$, et les résidus des positions p' n'ayant pas $p0$ comme préfixe sont $\{p'\}$. Par conséquent, toutes les positions de $\mathcal{P}$ ont exactement une position dans leur résidu, chaque ensemble de positions étant envoyé sur l'ensemble des positions où ces processus récursifs sont situés après la transition. Cela signifie que

$$\Omega \rhd \mathfrak{C}_M[\mathfrak{C}_T[(\!|\mathfrak{C}_?[l[\![c\,?\,(X:\mathsf{T})\,P]\!]]|\!)^{\varepsilon}_{\mathcal{P}}] \xrightarrow{(\Phi)c?V} \Omega;\Phi \rhd \mathfrak{C}_M[\mathfrak{C}_T[\mathfrak{C}'_?[l[\![(\!|P\{^V/_X\}|\!)^{p00}_{\mathcal{P}}]\!]]]]$$

avec, dans le cas où P n'est pas un processus récursif,

$$\mathfrak{C}_M[\mathfrak{C}_T[\mathfrak{C}'_?[l[\![(\!|P\{^V/_X\}|\!)^{p00}_{\mathcal{P}}]\!]]]] = {}_{sys}(\!|\mathfrak{C}_M[\mathfrak{C}_?[l[\![P\{^V/_X\}]\!]]]|\!)^{\varepsilon}_{\mathcal{P}'}$$

où $\mathcal{P}'$ est le résidu de $\mathcal{P}$ après cette transition. Dans le cas où P est un processus récursif de la forme $(\mathsf{rec}\ Z : \mathsf{R}(X)\,.\,P')\,\langle V\rangle$, il peut avoir deux traductions distinctes dans $(\!|M|\!)^{\varepsilon}_{\mathcal{P}}$:

$$(\!|(\mathsf{rec}\ Z : \mathsf{R}(X)\,.\,P')\,\langle V\rangle|\!)^{p00}_{\mathcal{P}} = \mathsf{here}\,(x)\,\mathsf{goto}\ ref_E.\ inst_E\,!\,\langle x, V\rangle$$

dans le cas où $p \in E \in \mathcal{P}$; ou

$$\begin{array}{l}(\!|(\mathsf{rec}\ Z : \mathsf{R}(X)\,.\,P')\,\langle V\rangle|\!)^{p00}_{\mathcal{P}} = \\ \quad \mathsf{newloc}\ ref_Z, inst_Z : \sum x : \textsc{loc}.\ \textsc{rw}\langle \mathsf{R}\rangle_{@}x\ \mathsf{with} \\ \qquad\qquad *inst_Z\,?\,((x_Z, X) : \mathsf{R})\,\mathsf{goto}\ x_Z.\ (\!|P'|\!)^{p000}_{\mathcal{P}}\ \mathsf{in}(\!|Z\,\langle V\rangle|\!)\end{array}$$

si la position p n'apparaît pas dans $\mathcal{P}$.
Dans le premier cas, on voit facilement que la traduction, quand elle est déclenchée dans la localité l, peut effectuer deux actions invisibles pour se réduire en $l[\![inst_E\,!\,\langle l, V\rangle]\!]$. Dans le second cas, dans la même localité, il peut se réduire en :

$$\begin{array}{l}(\mathsf{new}\ ref_Z : \textsc{loc})\,(\mathsf{new}\ inst_Z : \textsc{rw}\langle\rangle_{@}ref_Z) \\ \quad ref_Z[\![*inst_Z\,?\,((x_Z, X) : \mathsf{R})\,\mathsf{goto}\ x_Z.\ (\!|P'|\!)^{p000}_{\mathcal{P}}]\!] \\ \mid\ ref_Z[\![inst_Z\,?\,((x_Z, X) : \mathsf{R})\,\mathsf{goto}\ x_Z.\ (\!|P'|\!)^{p000}_{\mathcal{P}}]\!] \\ \mid\ ref_Z[\![inst_Z\,!\,\langle l, V\rangle]\!]\end{array}$$

Où que soit placé ce processus récursif, la structure sans new de M dans le système $\mathfrak{C}_M[M]$ permet de déduire que

$$\begin{array}{rl}\mathfrak{C}_M[\mathfrak{C}_T[\mathfrak{C}_p[l[\![(\!|(\mathsf{rec}\ Z : \mathsf{R}(X)\,.\,P')\,\langle V\rangle|\!)^{p00}_{\mathcal{P}}]\!]]] & \\ \xrightarrow{\tau}{}^*_{\beta} \equiv & \mathfrak{C}_M[\mathfrak{C}_T[(\mathsf{new}\ ref_Z : \textsc{loc})\,(\mathsf{new}\ inst_Z : \textsc{rw}\langle\rangle_{@}ref_Z) \\ & \quad ref_Z[\![inst_Z\,?\,((x_Z, X) : \mathsf{R})\,\mathsf{goto}\ x_Z.\ (\!|P'|\!)^{p0}_{\mathcal{P}}]\!] \\ & \quad \mid ref_Z[\![*inst_Z\,?\,((x_Z, X) : \mathsf{R})\,\mathsf{goto}\ x_Z.\ (\!|P'|\!)^{p0}_{\mathcal{P}}]\!] \\ & \quad \mid \mathfrak{C}_p[ref_Z[\![inst_Z\,!\,\langle l, V\rangle]\!]]]]\end{array}$$

On voit alors qu'il s'agit de la traduction habituelle d'un processus récursif. Il suffit en effet de rajouter $\{p0, p0p_1, \ldots, p0p_n\} : 0$ à la sous-partition, où les p_i sont les positions des appels récursifs dans le processus récursif.
Dans tous les cas, on a vu que

$$\Omega;\Phi \rhd \mathfrak{C}_M[\mathfrak{C}_T[\mathfrak{C}'_?[l[\![(\!|P\{^V/_X\}|\!)^{p00}_{\mathcal{P}}]\!]]] \xrightarrow{\tau}{}^*_{\beta} \equiv \Omega;\Phi \rhd {}_{sys}(\!|\mathfrak{C}_M[\mathfrak{C}_?[l[\![P\{^V/_X\}]\!]]]|\!)^{\varepsilon}_{\mathcal{P}''}$$

pour un certain $\mathcal{P}''$.

(te-w), (te-goto), (te-here) Ces cas sont similaires au précédent.

(te-split), (te-rep) À une récriture près des positions, ces cas aussi sont similaires au précédent.

(te-if-v) et (te-if-f) Si

$$M = \mathfrak{C}_{\mathsf{if}}[l[\![\mathsf{if}\ a_1 = a_2\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2]\!]]$$

où $\mathsf{if}\ a_1 = a_2\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2$ est à la position p dans M, on obtient alors

$$\begin{aligned}&{}_{sys}(\!|\mathfrak{C}_M[\mathfrak{C}_{\mathsf{if}}[l[\![\mathsf{if}\ a_1 = a_2\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2]\!]]]|\!)^{\varepsilon}_{\mathcal{P}}\\ &\quad = \mathfrak{C}_M[\mathfrak{C}_T[\mathfrak{C}_{\mathsf{if}}[l[\![\mathsf{if}\ a_1 = a_2\ \mathsf{then}\ (\!|P_1|\!)^{p01}_{\mathcal{P}}\ \mathsf{else}\ (\!|P_2|\!)^{p02}_{\mathcal{P}}]\!]]]]\end{aligned}$$

La transition de $\Omega \rhd \mathfrak{C}_M[M]$ peut être mimée, par exemple dans le cas (TE-IF-V), par :

$$\begin{aligned}\Omega \rhd \mathfrak{C}_M[\mathfrak{C}_T[\mathfrak{C}_{\mathsf{if}}[l[\![\mathsf{if}\ a_1 = a_2\ \mathsf{then}\ (\!|P_1|\!)^{p01}_{\mathcal{P}}\ \mathsf{else}\ (\!|P_2|\!)^{p02}_{\mathcal{P}}]\!]]]]\\ \xrightarrow{\tau}\Omega \rhd \mathfrak{C}_M[\mathfrak{C}_T[\mathfrak{C}_{\mathsf{if}}[l[\![(\!|P_1|\!)^{p01}_{\mathcal{P}}]\!]]]]\end{aligned}$$

Naturellement le résidu de toutes les positions de la forme $p01p_1$ sont les singletons $\{p0p_1\}$, les positions de la forme $p02p_2$ ont pour résidu l'ensemble vide, et, enfin, toutes les positions n'ayant pas $p0$ comme préfixe ont pour résidu le singleton qui les contient. Si $\mathcal{P}$ contient un ensemble E dont le résidu après cette transition est vide, cela signifie que tous les processus récursifs auxquels E correspond sont dans P_2. Les noms ref_E et $inst_E$ n'apparaissent donc pas dans $\mathfrak{C}_{\mathsf{if}}[l[\![(\!|P_1|\!)^{p01}_{\mathcal{P}}]\!]]$. La partie correspondante de $\mathfrak{C}_T$ est alors, d'après le lemme 3.13, fortement bisimilaire à $(\mathsf{new}\, l : \mathrm{LOC})\, l[\![*\,\mathsf{stop}]\!]$. En utilisant éventuellement un raisonnement similaire au cas précédent si P est un processus récursif, on sait alors que :

$$\Omega \rhd \mathfrak{C}_M[\mathfrak{C}_T[\mathfrak{C}_{\mathsf{if}}[l[\![(\!|P_1|\!)^{p01}_{\mathcal{P}}]\!]]]] \xrightarrow{\tau}{}^{*}_{\beta} \sim^t \Omega \rhd {}_{sys}(\!|\mathfrak{C}_M[\mathfrak{C}_T[\mathfrak{C}_{\mathsf{if}}[l[\![P_1]\!]]]]|\!)^{\varepsilon}_{\mathcal{P}'}$$

ou que :

$$\begin{aligned}&\Omega \rhd \mathfrak{C}_M[\mathfrak{C}_T[\mathfrak{C}_{\mathsf{if}}[l[\![(\!|P_1|\!)^{p01}_{\mathcal{P}}]\!]]]]\\ &\quad \xrightarrow{\tau}{}^{*}_{\beta} \sim^t \Omega \rhd {}_{sys}(\!|\mathfrak{C}_M[\mathfrak{C}_T[\mathfrak{C}_{\mathsf{if}}[l[\![P_1]\!]]]]|\!)^{\varepsilon}_{\mathcal{P}'} \mid (\mathsf{new}\, l : \mathrm{LOC})\, l[\![*\,\mathsf{stop}]\!]\end{aligned}$$

puisque la congruence structurelle est incluse dans la bisimilarité forte.

(te-newchan), (te-newloc) Il suffit dans ces cas de remarquer que la portée des nouveaux noms peut être étendue jusqu'au niveau du contexte $\mathfrak{C}_M$ par congruence structurelle. Cette extension repose partiellement sur le fait que les processus récursifs situés sous le préfixe newchan ou newloc ne peuvent mentionner les noms générés par ces préfixes. Mis à part cette remarque, ces cas sont similaires aux cas précédents.

(te-rec) La traduction garantit alors que la position $p0$ du processus récursif est telle que $p0 \in E \in \mathcal{P}$ et

$$(\!|l[\![(\mathsf{rec}\ Z : \mathsf{R}\,(X)\,.\,P)\,\langle V\rangle]\!]|\!)^{p}_{\mathcal{P}} = ref_E[\![inst_E\,!\,\langle l, V\rangle]\!]$$

Par ailleurs, la traduction garantit également l'existence d'un processus $ref_E[\![inst_E\,?\,(x_E, X : \mathsf{R})\cdots]\!]$ dans $\mathfrak{C}_T$. On simule alors le dépliage du processus récursif en effectuant la communication sur le canal $inst_E$. D'après

l'hypothèse émise sur la sous-partition $\mathcal{P}$ et par commutation entre la substitution et la traduction déjà vue pour (TE-R), on sait que la continuation de la réception sur ce canal est de la forme $ref_E[\![\mathsf{goto}\, l.\ (\!|P\{^V/_X\}|\!)^{p00}_{\mathcal{P}}]\!]$. Dans le cas où il y a dans la traduction de $\mathfrak{C}_M[M]$ une seule instance du processus répliqué correspondant au processus récursif, c'est-à-dire si $E : 0 \in \mathcal{P}$, il est nécessaire d'en générer une nouvelle pour rester dans la traduction, ce qui requiert seulement une β-réduction. Par ailleurs, par inclusion de la congruence structurelle dans la bisimilarité forte, on voit facilement que la continuation de la communication peut être replacée à la position p et y remplacer $ref_E[\![\mathsf{stop}]\!]$, qui est fortement bisimilaire à $\mathbf{0}$. Cette continuation peut, qui plus est, effectuer une β-réduction pour devenir $l[\![(\!|P\{^V/_X\}|\!)^{p00}_{\mathcal{P}}]\!]$. Or toutes les positions des appels récursifs $Z\,\langle V'\rangle$ dans P sont aussi dans l'ensemble E par définition de la sous-partition $\mathcal{P}$, ce qui implique que la traduction $P\{^V/_X\}$ et celle de $P\{^V/_X\}[^{\mathsf{rec}\ Z:\mathsf{R}(X).\ P}/_Z]$ coïncident directement. Qui plus est, les positions de toutes les variables de récursion autres que Z et de tous les processus récursifs mentionnés dans P se retrouvent aux positions de leur résidu. Par un raisonnement similaire au cas (TE-IF-V) si P ne contient aucun appel récursif $Z\,\langle V\rangle$ et que le résidu de E par cette transition est vide, on obtient alors :

$$\begin{array}{l}\Omega \rhd \mathfrak{C}_M[\mathfrak{C}_T[\mathfrak{C}_{\mathsf{rec}}[l[\![(\!|P\{^V/_X\}|\!)^{p00}_{\mathcal{P}}]\!]]]]\\ \quad \xrightarrow{\tau}{}^*_\beta \sim^t \Omega \rhd {}_{sys}(\!|\mathfrak{C}_M[\mathfrak{C}_T[\mathfrak{C}_{\mathsf{rec}}[l[\![P\{^V/_X\}[^{\mathsf{rec}\ Z:\mathsf{R}(X).\ P}/_Z]]\!]]]]|\!)_{\mathcal{P}'}\end{array}$$

ou :

$$\begin{array}{l}\Omega \rhd \mathfrak{C}_M[\mathfrak{C}_T[\mathfrak{C}_{\mathsf{rec}}[l[\![(\!|P\{^V/_X\}|\!)^{p00}_{\mathcal{P}}]\!]]]]\\ \quad \xrightarrow{\tau}{}^*_\beta \sim^t \Omega \rhd \begin{array}{l}{}_{sys}(\!|\mathfrak{C}_M[\mathfrak{C}_T[\mathfrak{C}_{\mathsf{rec}}[l[\![P\{^V/_X\}[^{\mathsf{rec}\ Z:\mathsf{R}(X).\ P}/_Z]]\!]]]]|\!)_{\mathcal{P}'}\\ \quad | \,(\mathsf{new}\, l : \mathrm{LOC})\, l[\![* \,\mathsf{stop}]\!]\end{array}\end{array}$$

où $\mathcal{P}'$ est soit le résidu de $\mathcal{P}$ après la transition du système $\mathfrak{C}_M[M]$ soit le résidu enrichi si P est un processus récursif non traduit, exactement comme dans les cas précédents.

Le cas de la communication qui nous reste à considérer se ramène simplement aux cas (TE-R) et (TE-W).

Enfin, envisageons que la configuration

$$\Omega \rhd {}_{sys}(\!|\mathfrak{C}_M[M]|\!)_{\mathcal{P}} \mid (\mathsf{new}\, l : \mathrm{LOC})\, l[\![* \,\mathsf{stop}]\!]$$

effectue une action. Si cette action est un déroulement du processus $* \,\mathsf{stop}$, le résultat est immédiat car $(\mathsf{new}\, l : \mathrm{LOC})\, l[\![\mathsf{stop}]\!] \mid l[\![* \,\mathsf{stop}]\!]$ est fortement bisimilaire à $(\mathsf{new}\, l : \mathrm{LOC})\, l[\![* \,\mathsf{stop}]\!]$. Si cette action est un déroulement du processus répliqué correspondant à l'ensemble de positions E situé dans le contexte $\mathfrak{C}_T$ introduit par la traduction, on sait que $\mathcal{P}$ doit être de la forme $\mathcal{P}_1, E : n, \mathcal{P}_2$ et que la transition est nécessairement :

$$\Omega \rhd {}_{sys}(\!|\mathfrak{C}_M[M]|\!)_{\mathcal{P}} \xrightarrow{\tau} \Omega \rhd {}_{sys}(\!|\mathfrak{C}_M[M]|\!)_{\mathcal{P}_1, E:(n+1), \mathcal{P}_2}$$

qui est facilement simulée par une absence de transition du système $\mathfrak{C}_M[M]$. S'il s'agit d'une communication sur un canal $inst_E$, le préfixe $inst_E\,!\,\langle l, V\rangle$ intervenant dans la communication est nécessairement la traduction d'un processus récursif $(\mathsf{rec}\ Z : \mathsf{R}\,(X)\,.\ P)\,\langle V\rangle$. L'adéquation de la communication avec le dépliage du

processus récursif modulo β-réductions et bisimilarité forte se prouve comme dans le cas où $\Omega \rhd \mathfrak{C}_M[M]$ effectue l'action. Toute autre action de la configuration traduite doit être effectuée par un ou deux préfixes de $(\!|M|\!)^{\varepsilon}_{\mathcal{P}}$ qui sont laissés invariants par la traduction. On peut donc prouver que cette action est mimée par la même preuve que dans le cas où $\Omega \rhd \mathfrak{C}_M[M]$ effectue l'action, ce qui achève la preuve. □

Ce lemme permet d'obtenir le résultat recherché :

Théorème 3.15 (Équivalence de la traduction)**.** *Supposons un système M et un environnement Ω tel que $\Omega \rhd M$ soit une configuration bien formée. Alors $\Omega \vDash M \cong^{tbr} (\!|M|\!)$.*

Démonstration. D'après la proposition 3.3, la relation $\Omega \vDash M \; \mathcal{R}_1 \; (\!|M|\!)$ est typée.

Il est possible de récrire M par congruence structurelle sous la forme $\mathfrak{C}_{M'}[M']$ où le contexte $\mathfrak{C}_{M'}$ est de la forme $(\mathsf{new}\, a_1 : \mathsf{E}_1) \cdots (\mathsf{new}\, a_n : \mathsf{E}_n)[\cdot]$ et M' est M dépourvu de ses new. La même opération peut être effectuée sur $(\!|M|\!)$ pour obtenir $\mathfrak{C}_{M'}[(\!|M'|\!)]$. Par transitivité de $\cong^{tbr}$, il suffira de prouver que la relation $\Omega \vDash \mathfrak{C}_{M'}[M'] \; \mathcal{R}_2 \; \mathfrak{C}_{M'}[(\!|M'|\!)]$ est incluse dans $\cong^{tbr}$.

Le théorème 2.45 de coïncidence des équivalences typées montre qu'il suffit de prouver que $\mathcal{R}_2$ est incluse dans $\approx^t$. Qui plus est, on voit facilement que $\mathfrak{C}_{M'}[(\!|M'|\!)]$ peut effectuer une série de $\xrightarrow{\tau}_{\beta}$ et utiliser la congruence structurelle pour aboutir à un système de la forme ${}_{sys}(\!|\mathfrak{C}_{M'}[M']|\!)_{\mathcal{P}}$, où toutes les localités-refuges correspondant aux processus récursifs de M' ont été engendrées. Il suffit donc de prouver que $\Omega \vDash \mathfrak{C}_{M'}[M'] \; \mathcal{R}_3 \; {}_{sys}(\!|\mathfrak{C}_{M'}[M']|\!)_{\mathcal{P}}$ est incluse dans $\approx^t$, puisque $\xrightarrow{\tau}_{\beta}$ est incluse dans la bisimilarité.

Enfin, les systèmes ${}_{sys}(\!|\mathfrak{C}_{M'}[M']|\!)_{\mathcal{P}}$ et ${}_{sys}(\!|\mathfrak{C}_{M'}[M']|\!)_{\mathcal{P}} \mid (\mathsf{new}\, l : \textsc{loc})\, l[\![* \,\mathsf{stop}]\!]$ étant bisimilaires, le lemme 3.14 permet de conclure. □

Chapitre 4

Typage de la mobilité

4.1 Là où le bât blesse

Le calcul typé présenté aux chapitres 1 et 2 laisse les processus migrer librement d'une localité à l'autre, par la simple invocation de la primitive de migration goto k. On voudrait pourtant permettre à une localité d'imposer une politique de contrôle sur les processus qu'elle accepte de laisser s'exécuter dans son domaine. Plusieurs travaux ont été effectués dans cette voie dans le cadre du Dπ-calcul. Dans [HMR03], une approche de la question est proposée en imposant qu'un processus possède une capacité associée à la localité dans laquelle il veut migrer. Dans [HRY05], les processus désirant pénétrer dans une localité doivent y connaître un *port*, auquel un jeu d'autorisations est associé : ces autorisations mentionnent l'ensemble des capacités que le processus aura le droit d'utiliser après son entrée dans la localité. Ce contrôle est donc particulièrement restrictif, et suppose un typage des processus afin d'obtenir un type majorant l'ensemble des capacités nécessaires au bon déroulement du processus après la migration. Par ailleurs, cette approche assimile les migrations à des actions d'ordre supérieur. On s'intéressera dans ce chapitre à une solution intermédiaire, qui vise à définir un contrôle potentiellement aussi puissant que celui de [HRY05] tout en restant du premier ordre.

D'autres possibilités ont été envisagées. On peut citer [MV05], dans lequel les politiques de sécurité des localités décrivent, par quelques capacités, les droits accordés aux processus entrants en fonction de la suite de localités par lesquelles ils sont passés préalablement. Le système que l'on propose ici ignorera systématiquement toutes les localités par lesquelles un processus est passé sauf la dernière, celle d'où il arrive. On suppose en effet que la provenance immédiate d'un processus peut être établie de façon fiable, ce qui semble raisonnable, ne serait-ce qu'en supposant la coopération de la localité en question, alors que la fiabilité du parcours suivi semble plus spéculative.

Ce nécessaire concours de la localité d'origine traduit le fait que le contrôle de provenance exprime les liens de *confiance* établis entre les localités. Ce principe de confiance entre localités est déjà apparu dans [RH03]. Sa représentation concrète sera cependant très différente ici : certains identifiants du langage joueront le rôle de *passeports*. Ces passeports devront être mentionnés lors de toute migration et ne seront valables que pour entrer dans une localité précise, en provenance

de certaines localités. On pourra donc les voir, en quelque sorte, comme des « mots de passe ». Afin de permettre la modélisation de « localité publique », universellement accessible, telle qu'un serveur par exemple, il existera aussi des *passeports universels*, autorisant l'accès depuis n'importe quelle localité.

L'objectif à moyen terme du contrôle par passeport est d'exiger aussi que les ressources auxquelles un processus peut accéder dans la localité où il entre correspondent au passeport dont il dispose pour entrer. Cette idée est très proche des intuitions de [HRY05], tout en offrant une théorie plus simple. Par ailleurs, suivant l'idée de confiance pouvant s'établir entre localités, on voudra pouvoir éviter de contrôler rigoureusement le comportement de processus provenant de « localités amies ». Le calcul combinerait alors deux grandes familles de passeports :

- les passeports avec contrôle strict des ressources accessibles après migration ;
- les passeports libéraux sur cette question ;

ces deux familles comportant encore deux sortes de passeports :

- les passeports avec contrôle strict des provenances ;
- les passeports libéraux sur cette question.

On étudiera extensivement dans la suite les passeports *sans* contrôle des ressources accessibles après migration, dans l'espoir que ce fragment, plus simple, révèle l'essentiel des difficultés techniques de la théorie complète. On esquissera dans la section 4.8 l'impact de passeports ouvrant accès aux ressources.

4.2 Présentation du calcul avec passeport

Afin de permettre à une localité de contrôler la provenance des processus qu'elle héberge, la construction $\mathsf{goto}\, k$ du calcul devra mentionner explicitement une autorisation : le processus

$$\mathsf{goto}_p\, k.\ P$$

invoque le *passeport* p pour demander à entrer dans la localité k et y exécuter sa continuation P. Par cette primitive, le processus qui veut migrer demande à la localité k d'autoriser cette migration eu égard au passeport p. De cette façon, k peut interdire l'accès à tout processus non dûment autorisé.

Cette vision simple des passeports permettra de les intégrer très naturellement au langage développé dans les chapitres précédents : ils seront représentés par des noms ou des variables, exactement comme les canaux et les localités. Ce choix garantira une grande homogénéité du calcul, tout en permettant de réutiliser la souplesse des communications de noms, notamment l'extension des portées, pour les passeports. En particulier, un processus client qui demande à un serveur situé dans la localité *sv* de calculer un résultat pourra s'écrire, en omettant pour l'instant les types :

$$cl[\![\mathsf{goto}_{p_{sv}}\, sv.\ req\,!\,\langle (sv, cl), (question, rés, pass)\rangle \mid \ldots rés\,?\,(x)\ P]\!]$$

Le processus peut en effet transmettre, en plus de sa *question* et du canal sur lequel le résultat doit être rendu (*rés* dans *cl*), un passeport *pass* à utiliser pour entrer dans la localité *cl* afin d'y communiquer le résultat. Le serveur en question ressemblerait à :

$$sv[\![*req\,?\,((x_{sv}, x_{cl}), (x_{question}, x_{rés}, x_{pass}) : \mathsf{T}) \cdots \textsc{calcul} \cdots \mathsf{goto}_{x_{pass}}\, x_{cl}.\ x_{rés}\,!\,\langle r\rangle]\!]$$

où le passeport utilisé à la fin du traitement de la requête est une variable qui est instanciée lorsqu'un client transmet sa requête. Le client contrôle ainsi explicitement, à la fois les localités auxquelles il donne accès mais également à quel instant et sur quel canal cette permission est réellement transmise.

Afin de permettre au client de donner au serveur accès à sa localité, on rajoutera au calcul la possibilité de générer de nouveaux passeports. Tout comme newloc permet de créer dynamiquement de nouvelles localités et newchan de nouveaux canaux, newpass introduira de nouveaux passeports. À l'aide de cette nouvelle construction, on peut enrichir l'exemple précédent de la façon suivante :

$$cl[\![\mathsf{newpass}\ pass\ \mathsf{in}\ \mathsf{goto}_{p_{sv}}\ sv.\ req!\,\langle(sv, cl), (question, rés, pass)\rangle \mid \ldots rés?\,(x)\,P]\!]$$

Le processus commence par générer le passeport *pass* avant de le communiquer au serveur pour l'autoriser à venir donner sa réponse. Il va de soi que le passeport ainsi créé ne peut servir qu'à accéder à la localité *cl* dans laquelle la primitive newpass est exécutée. Dans le cadre très simple du contrôle présenté ici, les processus qui auront le droit de créer un nouveau passeport pour entrer dans une localité donnée seront exactement les processus qui s'y exécutent. Cette hypothèse se justifie par le fait que tout processus ayant réussi à pénétrer dans la localité dispose d'un passeport idoine, il doit « donc » être fiable.

En revanche, cet exemple suggère une petite modification pour traiter cette interaction de façon plus stricte : le passeport *pass*, qui est délivré au serveur pour qu'il puisse rapporter son résultat, pourrait n'être généré que dans ce but-là en permettant de vérifier au passage que les processus l'utilisant proviennent effectivement du serveur. On exprimera cette nouvelle contrainte de la façon suivante :

$$\begin{array}{l} cl[\![\mathsf{newpass}\ pass\ \mathsf{from}\ sv\ \mathsf{in} \\ \qquad \mathsf{goto}_{p_{sv}}\ sv.\ req!\,\langle(sv, cl), (question, rés, pass)\rangle \mid \ldots rés?\,(x)\,P]\!] \end{array}$$

Ce passeport *pass* ne sera alors utilisable que pour les migrations entre les localités *sv* et *cl*.

Bien entendu, cette méthode de contrôle n'empêche absolument pas un serveur voyou d'utiliser le passeport qui lui est donné pour attaquer le client. Cependant une utilisation plus précautionneuse des passeports permet d'éviter de compromettre toute la localité *cl*. Un client peut ainsi se protéger en générant une nouvelle localité servant d'intermédiaire, en procédant de la façon suivante :

$$\begin{array}{ll} cl[\![& \mathsf{newloc}\ (sas, rés_{sas}, pass_{sas}, pass_{cl}) \\ & \mathsf{with}\ rés_{sas}?\,(x)\ \mathsf{goto}_{pass_{cl}}\ cl.\ rés!\,\langle x\rangle \\ & \mathsf{in} \quad \mathsf{goto}_{p_k}\ k.\ req!\,\langle(sv, sas), (question, rés_{sas}, pass_{sas})\rangle \\ & \quad \mid \quad \cdots rés?\,(x)\,P \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad]\!] \end{array}$$

Dans cette version, au lieu de confier directement un passeport autorisant l'accès à la localité *cl*, le client ne communique qu'un passeport pour une localité temporaire, un *sas* créé pour l'occasion. C'est pourquoi il émet la requête $(sv, sas), (question, rés_{sas}, pass_{sas})$ au lieu de $(sv, cl), (question, rés, pass)$. Il doit par conséquent aussi placer un processus dans cette localité temporaire pour relayer la réponse en provenance du serveur. Pour que ce relais puisse fonctionner, il aura bien sûr besoin d'un passeport pour rentrer dans la localité du client *cl*. C'est pourquoi, lorsque la localité *sas* est générée, un passeport $pass_{cl}$ est créé au

Fig. 4.1 Modifications et ajouts à la syntaxe des processus (voir figure 1.2)

$P ::=$	*Processus*
$\mathsf{goto}_v\ u.\ P$	Migration
$\mathsf{newloc}\, l, (\vec{c}), (\vec{p}), (\vec{q}) : \mathsf{L}\ \mathsf{with}\ P_l\ \mathsf{in}\ P$	Création de localité
$\mathsf{newpass}\, p\, \mathsf{from}\, \tilde{u}\, \mathsf{in}\, P$	Création de passeport
$\mathsf{newpass}\, p\, \mathsf{from}\, \star\, \mathsf{in}\, P$	Création de passeport universel

passage, en plus du canal $rés_{sas}$ de réponse et du passeport $pass_{sas}$ permettant d'accéder au *sas*. Il est indispensable que ces passeports soient créés en même temps que le *sas*, sans quoi il serait impossible de migrer entre ce *sas* et la localité *cl* du client.

Le fait que les passeports soient des identifiants comme les autres permettra de garantir que seules les deux continuations de la primitive newloc disposent de ce passeport. Il est par conséquent assez facile de se convaincre que, dans cet exemple, il n'y aura pas de fuite et que ce passeport restera inaccessible aux processus envoyés par le serveur et s'exécutant dans le *sas*. En procédant de cette façon et malgré la simplicité du modèle de sécurité, le client se prémunit d'un détournement du passeport qu'il transmet au serveur. En utilisant systématiquement cette technique, c'est-à-dire en ne délivrant jamais de passeport donnant accès directement à sa localité mais toujours à des localités temporaires, le client pourra ainsi garder le contrôle sur tous les processus qui rentrent dans sa localité principale.

En utilisant la même démarche que dans les chapitres précédents, l'usage d'un système de types permet d'effectuer une analyse statique des processus afin de savoir s'ils n'essayent pas de surpasser leurs droits. Dans ce cadre de contrôle des migrations, un surpassement supplémentaire devra être évité : la tentative par un processus d'entrer dans une localité sans disposer d'un passeport adéquat. Le système de types que nous allons voir ajoutera pour cela un nouveau type d'identifiant correspondant aux passeports. Cet ajout permettra aussi d'utiliser la technologie des équivalences observationnelles typées développée dans les chapitres précédents pour vérifier la fiabilité apportée par l'introduction des passeports : l'environnement Ω modélisant les connaissances de l'observateur pourra ainsi indiquer finement quelles sont les localités auxquelles il peut accéder en précisant l'ensemble des passeports à sa disposition pour effectuer ses observations.

4.3 Extensions de la syntaxe, de la sémantique et des types

La syntaxe du calcul n'est que très légèrement modifiée pour ajouter des passeports contrôlant les migrations. En particulier la syntaxe des identifiants, valeurs et motifs donnée figure 1.1 est intégralement conservée. Il en est de même pour la syntaxe des systèmes. Les modifications et ajouts à la syntaxe des processus sont résumés à la figure 4.1.

La première modification nécessaire est l'indication explicite du passeport utilisé lors d'une migration : la migration est alors notée $\mathsf{goto}_v\, u.\ P$ où v est l'identifiant du passeport rendant l'accès à la localité u possible depuis la localité courante. Les autres modifications de la syntaxe rendent possible la création de ces passeports.

La première façon de créer de nouveaux passeports est d'utiliser la primitive $\mathsf{newpass}\, p\, \mathsf{from}\, \tilde{u}$ qui crée un nouveau passeport, de nom p, permettant de venir des localités $\tilde{u}$. Le fait qu'il s'agisse d'un ensemble de localités permettra de modéliser la confiance existant au sein d'un sous-réseau, où le même passeport peut être utilisé quelle que soit la localité de départ. Bien entendu, pour que les passeports permettent effectivement un contrôle des migrations, il est indispensable que seuls les processus s'exécutant au sein d'une localité puissent créer un passeport en autorisant l'accès. C'est pourquoi la primitive permettant de générer un nouveau passeport se contente d'indiquer les localités d'origine où ce passeport pourra être utilisé, la destination sera toujours la localité courante. La réduction du préfixe $\mathsf{newpass}\, p\, \mathsf{from}\, \tilde{u}$ se chargera alors de prendre en compte la localité courante pour donner un type au passeport p.

Il sera aussi possible de créer des passeports autorisant l'entrée dans la localité quelle que soit la localité d'origine, en utilisant la primitive $\mathsf{newpass}\, p\, \mathsf{from}\, \star$, où $\star$ peut se lire comme « n'importe où ». Ces passeports seront particulièrement adaptés à la description de comportements de serveurs, c'est-à-dire de localités publiquement accessibles.

La dernière modification de la syntaxe concerne les créations de localité, afin de pouvoir créer des passeports en même temps que la localité. En effet, si on considère le système $l[\![\mathsf{newloc}\, k\, \mathsf{with}\, P_k\, \mathsf{in}\, P]\!]$, le processus P ne pourra pas migrer vers la nouvelle localité k sans disposer d'un passeport spécifique. De plus, P_k n'aura le droit de migrer vers la localité l que s'il connaît un passeport universel pour y entrer, puisqu'il ne peut pas encore exister de passeport mentionnant le nouveau nom de localité k. Afin d'éviter ces blocages, la génération de localités est généralisée pour prendre la forme

$$\mathsf{newloc}\, k, (c_1, \ldots), (p_1, \ldots), (q_1, \ldots) : \mathsf{L}\, \mathsf{with}\, P_k\, \mathsf{in}\, P$$

où k et les c_i, p_i et q_i sont de nouveaux noms. k est la nouvelle localité, $c_1, \ldots$ un ensemble de canaux situés dans k, $p_1, \ldots$ des passeports permettant d'entrer dans k et enfin $q_1, \ldots$ des passeports pour entrer dans l, si l est la localité dans laquelle k est créée. Ainsi, les passeports p_i peuvent autoriser des migrations de l vers k et les passeports q_j celles de k vers l. La création de ces passeports et de ces canaux en même temps que la localité permet par ailleurs que ces noms soient connus par le processus P. Cette fonctionnalité rend élégante l'écriture de l'exemple du *sas* que nous avons vu précédemment : les droits de migration entre le sas et le client peuvent être réduits au strict minimum (un passeport permettant de migrer de *sas* vers *cl*), tout en autorisant le client à envoyer directement sa requête au serveur car il dispose du nom du canal $rés_{sas}$ sur lequel la réponse doit être retournée et du passeport $pass_{sas}$ nécessaire pour le serveur.

L'ajout des passeports suppose bien entendu l'ajout de types pour cette nouvelle catégorie de noms et de variables. Puisque les types apparaissent syntaxiquement dans les systèmes, voyons les types associés aux passeports avant les nouvelles règles de la sémantique.

Les modifications de la syntaxe des pré-types sont données à la figure 4.2. À un passeport autorisant les migrations vers la localité l partant de l'une des

Fig. 4.2 Modifications des pré-types (voir 2.1)

$\mathsf{P} ::=$	*Types de passeports*
$\tilde{s} \mapsto t$	Passeport
$\star \mapsto t$	Passeport universel
$\mathsf{E} ::=$	*Types des identifiants*
$\textsc{loc}$	Localité
$\mathsf{C}_{@}s$	Canal localisé à s
P	Passeport
$\mathsf{L} ::=$	*Types déclaratifs de localité*
$\sum x : \textsc{loc}.\ \sum y : \textsc{loc}.\ (\mathsf{C}_{1@}y, \ldots), (\tilde{u}_1^\star \mapsto y, \ldots), (\tilde{v}_1^\star \mapsto x, \ldots)$	

localités de l'ensemble $\{k_1, \ldots, k_n\}$ on associera le type[1] $k_1, \ldots, k_n \mapsto l$. Pour un passeport universel, on utilisera un type $\star \mapsto l$. On dénotera cette nouvelle famille de types par la lettre P, famille qui s'ajoute à celles des types possibles pour les identifiants.

Par contre la modification des types déclaratifs de localité est plus complexe. Tout comme dans le cadre du chapitre 2, le type déclaré pour les localités devra permettre d'associer un type de la forme $\mathsf{C}_{@}l$ à tous les canaux qui sont créés en même temps que l, où l est la nouvelle localité. Mais on veut aussi spécifier le type à attribuer aux différents passeports, aussi bien ceux autorisant l'accès à la nouvelle localité l que ceux permettant d'accéder à la « localité-mère », c'est-à-dire la localité où newloc est exécuté. Suivant la même approche que pour les types des canaux, on peut facilement imposer que la localité à laquelle la première famille de passeports donne accès soit une variable liée qui sera instanciée par la localité en cours de création, par exemple en écrivant :

$$k, (c_1, \ldots), (p_1, \ldots) : \sum y : \textsc{loc}.\ (\mathsf{C}_{1@}y, \ldots), (\tilde{u}_1^\star \mapsto y, \ldots)$$

L'expansion attribuera alors un type de la forme attendue à chacun des passeports p_i. La notation $\tilde{u}_1^\star$ apparaissant dans le type $\tilde{u}_1^\star \mapsto y$ désignera uniformément soit $\tilde{u}_1$ soit $\star$. $\tilde{u}_1^\star \mapsto y$ désignera ainsi soit un type de la forme $\tilde{u}_1 \mapsto y$ soit un type de la forme $\star \mapsto y$.

Pour la seconde famille de passeports, q_i, donnant accès à la localité-mère, on adapte cette idée, de sorte que les types déclaratifs de localité auront la forme

$$\sum x : \textsc{loc}.\ \sum y : \textsc{loc}.\ (\mathsf{C}_{1@}y, \ldots), (\tilde{u}_1^\star \mapsto y, \ldots), (\tilde{v}_1^\star \mapsto x, \ldots)$$

où x sera instanciée par la localité-mère et y par la localité-fille nouvellement créée.

Même si les types déclaratifs de localité utilisent une somme dépendante pour la localité-mère et la localité-fille, ces deux localités auront un statut sémantique totalement différent dans cette construction. Les quelques règles qui doivent changer sont données figure 4.3. On y voit en particulier que la règle de réduction du préfixe newloc effectue immédiatement la substitution de la première variable

[1] Les k_i apparaissant dans le type forment un ensemble, pas une liste. $a, b \mapsto l$ et $b, a \mapsto l$ sont deux notations du même type.

Fig. 4.3 Modifications de la sémantique par réductions (voir figure 1.4)

(R-GOTO) $l[\![\mathsf{goto}_p\, k.\ P]\!] \longrightarrow k[\![P]\!]$

(R-NEWLOC) $l[\![\mathsf{newloc}\, k, (\vec{c}), (\vec{p}), (\vec{q}) : \sum x : \text{LOC}.\ \mathsf{T}\ \mathsf{with}\ P_k\ \mathsf{in}\ P]\!] \longrightarrow (\mathsf{new}\langle k, ((\vec{c}), (\vec{p}), (\vec{q})) : \mathsf{T}\{^{l}/_{x}\}\rangle)(k[\![P_k]\!] \mid l[\![P]\!])$

(R-NEWPASS) $l[\![\mathsf{newpass}\, p\, \mathsf{from}\, \tilde{k}^{\star}\ \mathsf{in}\ P]\!] \longrightarrow (\mathsf{new}\, p : \tilde{k}^{\star} \mapsto l)\, l[\![P]\!]$

Fig. 4.4 Règles inductives supplémentaires de sous-typage (voir 2.4)

(SR-PASS) $\dfrac{\tilde{s}' \subseteq \tilde{s}}{\Sigma \vdash \tilde{s} \mapsto t <: \tilde{s}' \mapsto t}$ (SR-PASS-*) $\Sigma \vdash \star \mapsto t <: \tilde{s}^{\star} \mapsto t$

liée. En effet on utilise une variable liée pour la localité-mère dans les types L uniquement pour garantir que les passeports (q_i) qui sont engendrés lors de cette réduction permettent d'accéder à la localité-mère, et non à une localité quelconque. Il ne faut pas pour autant que la sémantique lie le nom de la localité-mère, donc on ne peut pas utiliser la notation $(\mathsf{new}\langle l, (k, ((\tilde{c}), (\tilde{p}), (\tilde{q}))) : \mathsf{L}\rangle\, @l)$ qui correspondrait à une suite de new commençant par $(\mathsf{new}\, l : \text{LOC})$. C'est pourquoi la règle (R-NEWLOC) substitue immédiatement la première variable du type L par le nom de la localité l où la réduction a lieu.

La règle de réduction (R-GOTO) modifiée peut paraître un peu surprenante : le seul changement qu'elle subisse est l'annotation du passeport utilisé pour la migration, mais elle ne dépend pas de ce passeport. Exactement comme dans la réduction des communications

$$l[\![a\,!\,\langle V\rangle\, P_1]\!] \mid l[\![a\,?\,(X : \mathsf{T})\, P_2]\!] \longrightarrow l[\![P_1]\!] \mid l[\![P_2\{^{V}/_{X}\}]\!]$$

où la validité de la substitution $\{^{V}/_{X}\}$ est tout simplement ignorée, le nom du passeport employé pour effectuer une migration ne joue aucun rôle dans la réduction. Sa validité découlera en revanche du typage qui sera imposé sur les systèmes. Par ailleurs, il est malgré tout indispensable de mentionner explicitement le nom du passeport dans la primitive de migration pour que ce nom soit libre dans le processus migrant : c'est la condition *sine qua non* pour que les règles de la congruence structurelle puissent s'appliquer normalement. La dernière règle ajoutée à la sémantique qui sert à créer les passeports est sans surprise.

Les types de passeport rajoutés à la syntaxe ne modifient que superficiellement la structure de l'ensemble des types. Il est en particulier très facile de rajouter des règles de sous-typage et de bornes inférieure et supérieure qui les prennent en compte. De par la forte similitude entre les règles inductives avec la mémoire Σ données aux figures 2.4, 2.5 et 2.6, et les fonctions des figures 2.2 et 2.3 servant à les définir dans le cadre coinductif, on se contentera de définir le cas récursif. En particulier, ces règles seront des axiomes puisque les types de passeport sont des types de base. Les règles supplémentaires sont alors données aux figures 4.4 et 4.5.

Fig. 4.5 Règles inductives supplémentaires pour $\sqcap$ (voir 2.5) et $\sqcup$ (voir 2.6)

$$\text{(MEET-PASS)} \quad \Sigma \vdash \tilde{s} \mapsto t \sqcap \tilde{s}' \mapsto t = \tilde{s}, \tilde{s}' \mapsto t$$

$$\text{(MEET-PASS-*-G)} \quad \Sigma \vdash \star \mapsto t \sqcap \tilde{s}^\star \mapsto t = \star \mapsto t$$

$$\text{(MEET-PASS-*-D)} \quad \Sigma \vdash \tilde{s}^\star \mapsto t \sqcap \star \mapsto t = \star \mapsto t$$

$$\text{(JOIN-PASS)} \quad \frac{\tilde{s} \neq \emptyset \text{ et } \tilde{t} \cap \tilde{t}' = \emptyset}{\Sigma \vdash \tilde{s}, \tilde{t} \mapsto s \sqcup \tilde{s}, \tilde{t}' \mapsto s = \tilde{s} \mapsto s}$$

$$\text{(JOIN-PASS-*-G)} \quad \Sigma \vdash \star \mapsto s \sqcup \tilde{s} \mapsto s = \tilde{s} \mapsto s$$

$$\text{(JOIN-PASS-*-D)} \quad \Sigma \vdash \tilde{s} \mapsto s \sqcup \star \mapsto s = \tilde{s} \mapsto s$$

$$\text{(JOIN-PASS-*-GD)} \quad \Sigma \vdash \star \mapsto s \sqcup \star \mapsto s = \star \mapsto s$$

Ces nouveaux pré-types ne modifient par contre en rien la théorie des types. En particulier, la définition 2.4, listant les conditions auxquelles un pré-type est un type, est inchangée : les contraintes de fermetures des portées des identifiants s'appliquent aussi aux identifiants apparaissant dans ces nouveaux types, pour lesquels les définitions des noms, variables et variables de récursion libres sont aisément étendues. Par ailleurs, tous les résultats sur l'ordre $<:$ et les opérateurs $\sqcap$ et $\sqcup$ obtenus dans le cadre coinductif, notamment le théorème 2.17 de complétude sous condition, restent applicables sur ces types.

4.4 Règles de typage

Le typage des processus et des systèmes en présence de passeports se bâtit exactement comme le typage sans passeport : puisque les propriétés de l'ordre de sous-typage sont conservées avec les types de passeport, la formation des environnements et le typage des valeurs, processus et systèmes sont naturels. Ainsi, la définition 2.23 est étendue de la façon suivante pour prendre en compte les passeports :

Définition 4.1 (Compatibilité large)**.** *On dit que les types* T_1 *et* T_2 *sont* compatibles au sens large *dans l'un des trois cas suivants :*

- *ils sont compatibles ;*
- *ils sont tous les deux de la forme* $\mathsf{C}_i @ s_i$*, les types* C_i *sont compatibles et les* s_i *ne sont pas tous les deux des noms ;*
- *ils sont tous les deux de la forme* $\tilde{s}_i^\star \mapsto t_i$ *et les* t_i *ne sont pas tous les deux des noms.*

On peut alors définir les environnements bien formés en rajoutant la règle donnée à la figure 4.6.

En revanche, aucune règle supplémentaire n'est nécessaire dans le typage des valeurs (vu à la figure 2.8). En effet, à la différence des canaux qui sont liés à une localité donnée et pour lesquels il existe un type particulier abrégé en C quand ils sont situés dans la localité courante, les passeports ne sont pas liés à une seule localité. Puisque l'intérêt principal des types abrégés C est de simplifier les communications lorsqu'aucune migration n'est nécessaire, une mesure similaire

Fig. 4.6 Règle supplémentaire des environnements de typage (voir figure 2.7)

$$\text{(E-PASS)}\quad \frac{\begin{array}{c}\Gamma \vdash \mathbf{env}\\ \Gamma \vdash s : \text{LOC}\\ \Gamma \vdash \tilde{s} : \tilde{\text{LOC}}\end{array}}{\Gamma, u : \tilde{s}^\star \mapsto s \vdash \mathbf{env}} \quad \downarrow(\Gamma(u) \cup \{\tilde{s}^\star \mapsto s\})$$

Fig. 4.7 Cas supplémentaires de l'extension substitutive (voir figure 2.11)

$$\begin{array}{rcl}
[\Gamma, s : \tilde{s} \mapsto s_i]_{s_2=s_1} & = & [\Gamma]_{s_2=s_1}, s : \tilde{s} \mapsto s_1, s : \tilde{s} \mapsto s_2 \quad \text{si } (\tilde{s} \cup \{s\}) \cap \{s_1, s_2\} = \emptyset \\
[\Gamma, s : s_i, \tilde{s} \mapsto s']_{s_2=s_1} & = & [\Gamma]_{s_2=s_1}, s : s_1, s_2, \tilde{s} \mapsto s' \quad \text{si } \{s, s'\} \cap \{s_1, s_2\} = \emptyset \\
[\Gamma, s : s_i, \tilde{s} \mapsto s_j]_{s_2=s_1} & = & [\Gamma]_{s_2=s_1}, s : s_1, s_2, \tilde{s} \mapsto s_1, s : s_1, s_2, \tilde{s} \mapsto s_2 \quad \text{si } s \notin \{s_1, s_2\} \\
[\Gamma, s : \star \mapsto s_i]_{s_2=s_1} & = & [\Gamma]_{s_2=s_1}, s : \star \mapsto s_1, s : \star \mapsto s_2 \quad \text{si } s \notin \{s_1, s_2\}
\end{array}$$

pour les types de passeport semble moins intéressante. On utilisera donc toujours leur type complet $\tilde{s}^\star \mapsto t$.

Pour cette raison l'expansion des valeurs, définie figure 2.10, sera une simple identité sur les passeports, soit la règle additionnelle :

$$\langle s : \tilde{s}^\star \mapsto s' \rangle \circledast t = s : \tilde{s}^\star \mapsto s'$$

On complète aussi la définition de l'extension substitutive. Les cas supplémentaires, donnés figure 4.7, prennent en compte de façon naturelle les substitutions nécessaires dans le cas d'un passeport, c'est-à-dire en substituant les identifiants pouvant apparaître dans les sources ou la cible du passeport.

Comme au chapitre 2, ces quelques notions permettent de donner les règles de typage des processus. Les règles modifiées, (T-GOTO) et (T-NEWLOC), ainsi que la nouvelle (T-NEWPASS) sont formalisées figure 4.8. Ces différentes règles retranscrivent simplement au niveau du typage les intuitions véhiculées par la sémantique, en particulier dans la mesure où les deux opérations essentielles sont les transformations des types des règles de génération de noms (T-NEWLOC) et (T-NEWPASS) qui apparaissent déjà dans la sémantique. Le typage des systèmes, tout comme celui des valeurs, ne requiert aucune modification, même bénigne.

4.5 Propriétés du typage

Comme celui introduit au chapitre 2, le système de types permet d'assurer des propriétés sur les processus et systèmes bien typés. Les propriétés vues section 2.8 se transposent et s'enrichissent dans le cadre présenté ici. Par exemple, la proposition 2.24 qui assure que tout environnement de typage Γ apparaissant dans un jugement de typage prouvable doit être bien formé reste parfaitement valable dans ce nouveau cadre. La preuve en est d'ailleurs parfaitement similaire, puisque le typage du Dπ-calcul présenté au chapitre 2 a soigneusement été conçu pour rendre naturelle l'extension que l'on donne ici. De la même façon, les notions de sous-typage et équivalence d'environnements de typage sont définies exactement de la même façon que précédemment (voir définition 2.25).

Fig. 4.8 Modifications et ajout au typage des processus (voir 2.9)

$$\text{(T-GOTO)}\quad \frac{\Gamma \vdash u : s \mapsto v \qquad \Gamma \mid \Delta \vdash_v P}{\Gamma \mid \Delta \vdash_s \mathsf{goto}_u\, v.\ P}$$

$$\text{(T-NEWLOC)}\quad \frac{\Gamma;\langle(k,((\vec{c}),(\vec{p}),(\vec{q}))) : \mathsf{T}\{^s/_x\}\rangle \mid \Delta \vdash_k P_k \qquad \Gamma;\langle(k,((\vec{c}),(\vec{p}),(\vec{q}))) : \mathsf{T}\{^s/_x\}\rangle \mid \Delta \vdash_s P}{\Gamma \mid \Delta \vdash_s \mathsf{newloc}\, k,(\vec{c}),(\vec{p}),(\vec{q}) : \sum x : \text{LOC}.\ \mathsf{T}\ \mathsf{with}\ P_k\ \mathsf{in}\ P}$$

$$\text{(T-NEWPASS)}\quad \frac{\Gamma \vdash \tilde{u} : \text{L}\tilde{\text{O}}\text{C} \qquad \Gamma; p : \tilde{u}^\star \mapsto s \mid \Delta \vdash_s P}{\Gamma \mid \Delta \vdash_s \mathsf{newpass}\, p\, \mathsf{from}\, \tilde{u}^\star\, \mathsf{in}\, P}$$

La propriété de sûreté du typage n'a pas été formalisée au chapitre 2, en faisant référence à la littérature sur le sujet. Les passeports, en revanche, sont totalement nouveaux et la fonction essentielle du typage de ces passeports est de garantir qu'un processus bien typé ne tente jamais de migrer sans disposer d'un passeport adéquat. Examinons donc comment formaliser la sûreté du typage des passeports. On définit pour cela une *réduction erronée* d'un système M, notée $M \xrightarrow{\text{err}}_\Gamma$, de la façon suivante :

$$\begin{aligned} l[\![\mathsf{goto}_p\, k.\ P]\!] &\xrightarrow{\text{err}}_\Gamma && \text{si } \Gamma \not\vdash_l p : l \mapsto k \\ M_1 \mid M_2 &\xrightarrow{\text{err}}_\Gamma && \text{si } M_1 \xrightarrow{\text{err}}_\Gamma \text{ ou } M_2 \xrightarrow{\text{err}}_\Gamma \\ (\mathsf{new}\, a : \mathsf{E})\, M &\xrightarrow{\text{err}}_\Gamma && \text{si } M \xrightarrow{\text{err}}_{\Gamma;a:\mathsf{E}} \end{aligned}$$

Proposition 4.2 (Sûreté du typage des passeports)**.** *Si* $\Gamma \vdash M$ *alors* $M \not\xrightarrow{\text{err}}_\Gamma$.

Démonstration. Supposons $M \xrightarrow{\text{err}}_\Gamma$ et prouvons que $\Gamma \not\vdash M$. M ne peut prendre que l'une des trois formes de systèmes apparaissant dans la définition de $\xrightarrow{\text{err}}_\Gamma$. S'il s'agit de $l[\![\mathsf{goto}_p\, k.\ P]\!]$, on en déduit que $\Gamma \not\vdash_l p : l \mapsto k$. Cela entraîne immédiatement que $\Gamma \not\vdash l[\![\mathsf{goto}_p\, k.\ P]\!]$. Les autres cas sont des simples applications de l'hypothèse d'induction. □

La propriété de sûreté du typage dans sa totalité, portant aussi sur les autres primitives du langage, se formalise de façon très similaire. Mais le typage assure de nombreuses autres propriétés dont on détaillera les preuves dans la suite. Le théorème fondamental assurant la préservation du typage restera malgré tout l'objectif premier de cet ensemble de propriétés. La preuve de ce théorème peut se décomposer en une série de lemmes et autres propositions.

1. Les lemmes les plus emblématiques de la préservation du typage portent sur les substitutions, qu'il s'agisse de traiter les appels récursifs (voir le lemme 4.10) ou bien les substitutions de variables (voir les lemmes 4.7 et 4.9).
2. Ces lemmes reposent eux-mêmes sur la propriété d'affaiblissement (proposition 4.3, déjà vue comme proposition 2.26).
3. Il est nécessaire de montrer que la congruence structurelle est cohérente avec le typage puisque la sémantique considère les systèmes à congruence

structurelle près. Cette preuve repose elle-même sur l'affaiblissement et son dual, le renforcement.

4. Enfin, d'autres lemmes traitent les modifications de l'environnement de typage par des extensions substitutives.

Pour attaquer ces propriétés dans l'ordre logique, nous allons voir successivement les points 2, 3, 1 et enfin 4.

Commençons donc par prouver la propriété d'affaiblissement qui donne tout son sens au sous-typage en assurant que tout ce qui est prouvable dans un environnement est prouvable dans ses environnements sous-types. Cette preuve repose elle-même sur un lemme à propos des extensions substitutives : pour prouver l'affaiblissement sur le typage des processus par induction, il faudra en effet que l'hypothèse de sous-typage et les extensions substitutives de (T-IF-V) commutent. Ce lemme 4.8 ne sera prouvé qu'ultérieurement : il est en effet considérablement généralisé afin de le combiner aussi avec des substitutions. Seul le fragment portant sur les extensions substitutives est nécessaire pour prouver l'affaiblissement. Cette généralisation sera en revanche utilisée ultérieurement dans le lemme 4.9 de substitution dans les processus, pour la même raison qui le fait apparaître dans l'affaiblissement.

Proposition 4.3 (Affaiblissement)**.** *Soient* Γ_1 *et* Γ_2 *deux environnements bien formés tels que* $\Gamma_1 <: \Gamma_2$*. Alors :*

1. $\Gamma_2 \vdash s : \mathsf{E}$ *implique* $\Gamma_1 \vdash s : \mathsf{E}$ *;*
2. $\Gamma_2 \vdash_s V : \mathsf{T}$ *implique* $\Gamma_1 \vdash_s V : \mathsf{T}$ *;*
3. $\Gamma_2 \mid \Delta \vdash_s P$ *implique* $\Gamma_1 \mid \Delta; \Delta' \vdash_s P$ *quel que soit* Δ' *;*
4. $\Gamma_2 \vdash M$ *implique* $\Gamma_1 \vdash M$*.*

Démonstration. Bien entendu ces propriétés se prouvent par induction sur la taille de la preuve de leur hypothèse en raisonnant sur la dernière règle utilisée.

1. – (V-ID) : $s : \mathsf{T}$ est une hypothèse de l'environnement Γ_2 ce qui implique par définition de $<:$ sur les environnements que $\Gamma_1 \vdash s : \mathsf{T}$.
   – (V-INF) : $\Gamma_1 \vdash s : \mathsf{T}$. On sait aussi qu'il existe T_1 et T_2 tels que $\Gamma_1 \vdash s : \mathsf{T}_1$, $\Gamma_1 \vdash s : \mathsf{T}_2$ et $\mathsf{T}_1 \sqcap \mathsf{T}_2 <: \mathsf{T}$. L'hypothèse d'induction permet de déduire immédiatement par (V-INF) que $\Gamma_2 \vdash s : \mathsf{T}$.
2. – (V-LOCALISATION) et (V-CANAL-LOC) se prouvent simplement en utilisant la propriété précédente.
   – (V-TUPLE) et (V-DEP) se prouvent directement par l'hypothèse d'induction.
3. Pour prouver la propriété sur les processus, on doit tout d'abord remarquer que $\Gamma_1 <: \Gamma_2$, $\mathsf{dom}(\Gamma_1) \cap \mathsf{dom}(\Gamma) = \emptyset$ et $\Gamma_2; \Gamma \vdash \mathbf{env}$ implique $\Gamma_1; \Gamma \vdash \mathbf{env}$ et $\Gamma_1; \Gamma <: \Gamma_2; \Gamma$ quel que soit Γ. Ces hypothèses permettent en effet de conclure que $\Gamma_1; \Gamma \vdash \mathbf{env}$ en procédant par une induction simple sur la taille de l'environnement Γ. Étant donné que les domaines de Γ_1 et Γ sont supposés distincts, les seuls identifiants mentionnés dans Γ qui ne sont pas définis uniquement dans Γ ne peuvent être que des localités qui doivent dès lors être définies dans Γ_2 car $\Gamma_2; \Gamma \vdash \mathbf{env}$. On conclut donc aisément de $\Gamma_1 <: \Gamma_2$ le fait que $\Gamma_1; \Gamma$ doit être bien formé.

   On obtient aussi facilement le résultat $\Gamma_1; \Gamma <: \Gamma_2; \Gamma$ en considérant une hypothèse $s : \mathsf{T}$ de $\Gamma_2; \Gamma$ et en raisonnant suivant son apparition dans Γ_2 ou dans Γ, les deux cas permettant de conclure rapidement $\Gamma_1; \Gamma \vdash s : \mathsf{T}$.

Ces remarques et l'affaiblissement dans le cas valeur permettent de traiter par induction toutes les règles de typage des processus sauf (T-IF). Pour cette règle, on suppose $\Gamma_2 \mid \Delta \vdash_s \mathsf{if}\ s_1 = s_2\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2$. Le lemme 4.8 appliqué avec la substitution identité permet de conclure que $\Gamma_1 <: \Gamma_2$ implique que si $[\Gamma_1]_{s_1=s_2} \vdash \mathbf{env}$ alors $[\Gamma_2]_{s_1=s_2} \vdash \mathbf{env}$ et $[\Gamma_1]_{s_1=s_2} <: [\Gamma_2]_{s_1=s_2}$. Dès lors une application de l'hypothèse d'induction permet de conclure que $[\Gamma_1]_{s_1=s_2} \mid \Delta; \Delta' \vdash_s P_1$ si $[\Gamma_1]_{s_1=s_2} \vdash \mathbf{env}$. Le résultat correspondant pour P_2 est une simple application de l'hypothèse d'induction, et on peut alors directement conclure que $\Gamma_1 \mid \Delta; \Delta' \vdash_s \mathsf{if}\ s_1 = s_2\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2$.

4. Ce résultat se prouve par une induction facile en utilisant l'affaiblissement des valeurs et processus et la même remarque que pour les processus, à savoir $\Gamma_1; \Gamma <: \Gamma_2; \Gamma$, pour traiter (T-NEW).

□

Au même titre que la propriété d'affaiblissement énoncée dans le cadre du chapitre 2 avec son corollaire 2.27, la proposition suivante a pour conséquence directe le fait que deux environnements équivalents admettent les mêmes jugements de typage.

Cette proposition a son dual. Bien entendu, il est impossible de conclure que $\Gamma_1 \vdash M$ dès que $\Gamma_2 \vdash M$ si $\Gamma_2 <: \Gamma_1$ dans le cas général, sans quoi le typage ne pourrait pas caractériser des comportements intéressants. Par contre, cette propriété est vraie dans le cas particulier où Γ_2 est un environnement sous-type uniquement parce qu'il contient des informations inutiles au typage.

Proposition 4.4 (Renforcement)**.** *Soient* Γ, s, E *tels que* $\Gamma; s : \mathsf{E}$ *soit bien formé. Alors :*

1. $\Gamma; s : \mathsf{E} \vdash s_0 : \mathsf{E}_0$ *et* $s \notin \mathsf{fn}(s_0 : \mathsf{E}_0)$ *impliquent* $\Gamma \vdash s_0 : \mathsf{E}_0$ *;*
2. $\Gamma; s : \mathsf{E} \vdash_{s_0} V : \mathsf{T}$ *et* $s \notin \{s_0\} \cup \mathsf{fn}(V : \mathsf{T})$ *impliquent* $\Gamma \vdash_{s_0} V : \mathsf{T}$ *;*
3. $\Gamma; s : \mathsf{E} \mid \Delta; Z_1 : \mathsf{R}_1, \ldots \vdash_{s_0} P$, $s \notin \{s_0\} \cup \mathsf{fn}(P)$ *et* $\{(Z_i)_i\} \cap \mathsf{frv}(P) = \emptyset$ *impliquent* $\Gamma \mid \Delta \vdash_{s_0} P$ *;*
4. $\Gamma; s : \mathsf{E} \vdash M$ *et* $s \notin \mathsf{fn}(M)$ *impliquent* $\Gamma \vdash M$*.*

Démonstration. Une induction sur l'hypothèse de typage suffit à prouver chacun des quatre cas. □

L'affaiblissement et le renforcement permettent de prouver que la congruence structurelle ne met en relation que des processus qui sont typables sous les mêmes hypothèses. En effet, on voit aisément que la règle de la congruence structurelle qui affirme $M_1 \mid (\mathsf{new}\, a : \mathsf{E})\, M_2 \equiv (\mathsf{new}\, a : \mathsf{E})(M_1 \mid M_2)$ repose sur le fait que $\Gamma \vdash M_1$ si et seulement si $\Gamma; a : \mathsf{E} \vdash M_1$ quand $a \notin \mathsf{fn}(M_1)$. Par ailleurs, rappelons que la sémantique fournie figure 4.3 identifie totalement les systèmes structurellement congrus. Il est donc primordial que le typage ne fasse pas de différence entre deux tels systèmes, c'est-à-dire qu'ils soient bien typés dans exactement les mêmes environnements.

Proposition 4.5 (Cohérence du typage avec la congruence structurelle)**.** *Soient deux systèmes* M *et* N *tels que* $M \equiv N$*. Quel que soit l'environnement* Γ*,* $\Gamma \vdash M$ *si et seulement si* $\Gamma \vdash N$*.*

Démonstration. Cette démonstration s'effectue par induction sur la preuve de l'équivalence $M \equiv N$. La réflexivité donne immédiatement le résultat, la symétrie et la transitivité ne nécessitent qu'une simple application de l'hypothèse d'induction. Le cas de contextualité suppose qu'il existe un contexte $\mathfrak{C}$ et deux systèmes M' et N' tels que $M = \mathfrak{C}[M']$ et $N = \mathfrak{C}[N']$. Par ailleurs, une simple analyse de la forme des contextes montre que, quel que soit O, si $\Gamma \vdash \mathfrak{C}[O]$ est prouvable, sa preuve passe par une preuve de $\Gamma; \Gamma' \vdash O$ pour un certain Γ' et pour tout O' tel que $\Gamma; \Gamma' \vdash O'$ on peut alors déduire $\Gamma \vdash \mathfrak{C}[O']$. $\Gamma \vdash M$ implique donc $\Gamma; \Gamma' \vdash M'$ pour un certain Γ' et l'hypothèse d'induction permet de conclure que $\Gamma; \Gamma' \vdash N'$ puis $\Gamma \vdash N$, et réciproquement.

Les autres cas correspondent aux règles de la congruence structurelle données à la figure 1.3. Regardons deux cas mettant en jeu les idées marquantes.

- Si $M = N \mid \mathbf{0}$, $\Gamma \vdash M$ implique immédiatement $\Gamma \vdash N$. La réciproque est fournie par la proposition 2.24, qui assure que $\Gamma \vdash N$ implique $\Gamma \vdash \mathbf{env}$ donc $\Gamma \vdash \mathbf{0}$.
- Si $M = M_1 \mid (\mathsf{new}\, a : \mathsf{E})\, M_2$ et $N = (\mathsf{new}\, a : \mathsf{E})(M_1 \mid M_2)$ avec $a \notin \mathsf{fn}(M_1)$, $\Gamma \vdash M$ implique $\Gamma \vdash M_1$. La proposition 4.3 permet de conclure que $\Gamma; a : \mathsf{E} \vdash M_1$. La réciproque découle de la proposition 4.4.

□

Concentrons-nous maintenant sur les lemmes de substitution annoncés pour prouver la préservation du typage. En effet, lorsqu'une communication prend place sur le canal c, le processus $c\,?\,(X : \mathsf{T})\, P$ se réduit en $P\{^V/_X\}$. Il est par conséquent indispensable de prouver que l'hypothèse de typage de P, à savoir $\Gamma; \langle X : \mathsf{T}\rangle \circledast l \mid \vdash_\iota P$, est suffisante pour conclure que $\Gamma \mid \vdash_\iota P\{^V/_X\}$. Bien entendu cette substitution sera possible uniquement parce que la valeur V aura au moins le type T dans Γ. Par rapport à la littérature existante sur les substitutions, les énoncés des propriétés et les preuves sont compliqués par l'indispensable traitement des extensions substitutives. Dans les systèmes de typage de la littérature, tous les environnements apparaissant dans la preuve de jugements de la forme $\Gamma, \langle X : \mathsf{T}\rangle \circledast l \vdash_\iota P$ sont eux-mêmes de la forme $\Gamma, \langle X : \mathsf{T}\rangle \circledast l, \Gamma'$. Mais l'extension substitutive brise cette uniformité. Par conséquent, là où la substitution sur les processus se prouve généralement en passant d'un environnement $\Gamma, \langle X : \mathsf{T}\rangle \circledast l, \Gamma'$ à $\Gamma, \Gamma'\{^V/_X\}$, on généralisera le lien qui existe entre les deux environnements mis en jeu.

Le premier lemme de substitution exprime le simple fait qu'associer un même type à une valeur et à un motif implique que les identifiants et les variables de base qui les constituent ont aussi des types appariés.

Lemme 4.6 (Substitution sur l'expansion des motifs)**.** *Soient* Γ*,* V*,* X*,* T*, et* s *tels que*

- $\Gamma \vdash_s V : \mathsf{T}$ *;*
- $\langle X : \mathsf{T}\rangle \circledast s = x_1 : \mathsf{T}_1, \ldots, x_n : \mathsf{T}_n$ *;*
- $\mathsf{fv}(X) \cap \mathsf{dom}(\Gamma) = \emptyset$*.*

Alors quel que soit i *compris entre* 1 *et* n*,* $\Gamma \vdash x_i\{^V/_X\} : \mathsf{T}_i\{^V/_X\}$*.*

Démonstration. On procède par induction sur la forme de T, qui force la façon dont l'expansion de $\langle X : \mathsf{T}\rangle \circledast s$ est effectuée. La preuve décompose ainsi progressivement T, X et V dans leurs différents composants, ce qui montre au passage que la décomposition arborescente simultanée de X et V existe. L'hypothèse d'induction utilisée pour cela ne peut être simplement celle de l'énoncé, dans

la mesure où le type T peut être dépendant : si le type T_{i+1} mentionne des identifiants parmi $x_1, \ldots, x_i$, v_{i+1} devra avoir le type $\mathsf{T}_{i+1}\{^{v_1}/_{x_1}\}\cdots\{^{v_i}/_{x_i}\}$. La propriété que l'on prouve par induction est donc :

- $V' = X'\{^{V}/_{X}\}$,
- $\Gamma \vdash_s V' : \mathsf{T}'\{^{V}/_{X}\}$,
- et $\langle X' : \mathsf{T}' \rangle \, @s = x'_1 : \mathsf{T}'_1, \ldots, x'_p : \mathsf{T}'_p$

implique que $\Gamma \vdash x'_i\{^{V}/_{X}\} : \mathsf{T}'_i\{^{V}/_{X}\}$.

On étudie donc les formes possibles pour T'.

- $\mathsf{T}' = \mathsf{C}$: l'expansion $\langle X' : \mathsf{C} \rangle \, @s$ est alors simplement de la forme $x' : \mathsf{C}_{@s}$. On note v' la valeur correspondant à x', c'est-à-dire $v' = x'\{^{V}/_{X}\}$. On sait donc $\Gamma \vdash_s v' : \mathsf{C}$ car le type C est clos. Ce jugement ne peut être prouvé que par (V-CANAL-LOC) ce qui implique $\Gamma \vdash v' : \mathsf{C}_{@s}$. La substitution laissant C et s invariants, on a le résultat $\Gamma \vdash x'\{^{V}/_{X}\} : \mathsf{C}_{@s}\{^{V}/_{X}\}$.
- $\mathsf{T}' = \text{LOC}$, $\mathsf{T}' = \mathsf{C}_{@t}$ et $\mathsf{T}' = \tilde{t}^\star \mapsto t$: ces cas aussi sont directs.
- $\mathsf{T}' = \sum \tilde{s} : \text{LÕC}.\ \mathsf{T}''$, alors X' est de la forme $(\tilde{x}, X'')$ et V' de la forme $(\tilde{v}, V'')$. $\Gamma \vdash_s V' : \mathsf{T}'\{^{V}/_{X}\}$ implique $\Gamma \vdash_s v_i : \text{LOC}$ pour tout i et $\Gamma \vdash_s V'' : \mathsf{T}''\{^{V}/_{X}\}\{^{\tilde{v}}/_{\tilde{s}}\}$ avec $\tilde{s} \cap X = \emptyset$ et $\tilde{s} \cap V = \emptyset$. On déduit alors $\Gamma \vdash_s V'' : \mathsf{T}''\{^{\tilde{v}}/_{\tilde{s}}\}\{^{V}/_{X}\}$ par commutation des substitutions[2], c'est-à-dire $\Gamma \vdash_s V'' : \mathsf{T}''\{^{\tilde{x}}/_{\tilde{s}}\}\{^{V}/_{X}\}$.
  Donc pour tous les x'_i dans $\tilde{x}$, le résultat est déjà obtenu, pour ceux de X'', on peut appliquer l'hypothèse d'induction.
- $\mathsf{T}' = (\mathsf{T}'_1, \ldots, \mathsf{T}'_p) : X'$ doit donc être de la forme $(X''_1, \ldots, X''_p)$ et V' de la forme $(V''_1, \ldots, V''_p)$. $V' = X'\{^{V}/_{X}\}$ permet de conclure que $V''_j = X''_j\{^{V}/_{X}\}$. De façon similaire, on obtient $\Gamma \vdash_s V''_j : \mathsf{T}''_j\{^{V}/_{X}\}$. L'hypothèse d'induction permet de conclure.

□

Prouvons maintenant les lemmes de substitution de valeurs proprement dits. Ces lemmes affirment que, s'il est possible de typer une valeur ou un processus dans un environnement Γ_1 et que l'environnement Γ_2 est, *grosso modo*, l'environnement Γ_1 après substitution, alors Γ_2 permet de typer les mêmes valeurs et processus après substitution. Comme évoqué précédemment, cette formulation est généralisée suffisamment pour interagir avec les extensions substitutives.

Lemme 4.7 (Substitution dans les valeurs)**.** *Soient Γ_1 et Γ_2 deux environnements, S_1 un motif pouvant contenir variables et variables de récursion et S_2 une valeur tels que pour tout $s : \mathsf{E}$ dans Γ_1, $\Gamma_2 \vdash s\{^{S_2}/_{S_1}\} : \mathsf{E}\{^{S_2}/_{S_1}\}$ soit prouvable. Alors :*

- *$\Gamma_1 \vdash t : \mathsf{E}$ implique $\Gamma_2 \vdash (t : \mathsf{E})\{^{S_2}/_{S_1}\}$;*
- *$\Gamma_1 \vdash_t V : \mathsf{T}$ implique $\Gamma_2 \vdash_{t\{^{S_2}/_{S_1}\}} (V : \mathsf{T})\{^{S_2}/_{S_1}\}$.*

Démonstration. Cette preuve s'effectue bien entendu par induction sur la taille de la preuve de $\Gamma_1 \vdash t : \mathsf{E}$.

- (V-ID) : si $\Gamma_1 \vdash t : \mathsf{E}$ est prouvé par (V-ID), $t : \mathsf{E}$ est directement une hypothèse de Γ_1 donc une application directe de l'hypothèse donne $\Gamma_2 \vdash (t : \mathsf{E})\{^{S_2}/_{S_1}\}$.
- (V-INF) : par hypothèse d'induction on obtient $\Gamma_2 \vdash (t : \mathsf{E}_1)\{^{S_2}/_{S_1}\}$ et $\Gamma_2 \vdash (t : \mathsf{E}_2)\{^{S_2}/_{S_1}\}$. Étant donné que la substitution est compatible avec $\sqcap$ car les types transmissibles par canaux sont invariants, $\mathsf{E}_1 \sqcap \mathsf{E}_2 = \mathsf{E}_3$

[2] voir la note 1, page 91

implique $(\mathsf{E}_1 \sqcap \mathsf{E}_2 = \mathsf{E}_3)\{^{S_2}/_{S_1}\}$. (V-INF) permet alors de conclure $\Gamma_2 \vdash (t : \mathsf{E}_3)\{^{S_2}/_{S_1}\}$.

La preuve pour les règles (V-LOCALISATION), (V-CANAL-LOC), (V-UPLET) et (V-DEP) se résume à des applications de l'hypothèse d'induction sur la preuve, en utilisant la commutation des substitutions dans le cas (V-DEP) puisque les variables liées $\tilde{x}$ sont conventionnellement choisies distinctes de toutes les variables apparaissant dans S_1 et S_2 par α-conversion. □

Lemme 4.8 (Le sous-typage à substitution près passe sous les extensions substitutives)**.** *Soient Γ_1 et Γ_2 deux environnements, S_1 un motif pouvant contenir variables et variables de récursion et S_2 une valeur tels que pour tout $s : \mathsf{E}$ dans Γ_1, $\Gamma_2 \vdash s\{^{S_2}/_{S_1}\} : \mathsf{E}\{^{S_2}/_{S_1}\}$ soit prouvable. Soient t_1 et t_2 deux identifiants. Alors, si $[\Gamma_2]_{t_1\{^{S_2}/_{S_1}\}=t_2\{^{S_2}/_{S_1}\}} \vdash \mathbf{env}$, $[\Gamma_1]_{t_1=t_2} \vdash \mathbf{env}$ et pour toute hypothèse $s : \mathsf{E}$ de $[\Gamma_1]_{t_1=t_2}$, $[\Gamma_2]_{t_1\{^{S_2}/_{S_1}\}=t_2\{^{S_2}/_{S_1}\}} \vdash (s : \mathsf{E})\{^{S_2}/_{S_1}\}$ est prouvable.*

Démonstration. On commence par prouver que la bonne formation de l'environnement $[\Gamma_2]_{t_1\{^{S_2}/_{S_1}\}=t_2\{^{S_2}/_{S_1}\}}$ implique celle de $[\Gamma_1]_{t_1=t_2}$ par contraposée.

On suppose $[\Gamma_1]_{t_1=t_2} \not\vdash \mathbf{env}$. Alors $[\Gamma_1]_{t_1=t_2}$ est de la forme $\Gamma, s : \mathsf{E}, \Gamma'$ où $\Gamma \vdash \mathbf{env}$ mais $\Gamma, s : \mathsf{E} \not\vdash \mathbf{env}$. On peut distinguer deux cas de mauvaise formation d'un environnement.

1. s est déjà défini dans l'environnement Γ avec $\not\downarrow(\Gamma(s) \cup \{\mathsf{E}\})$. On choisit donc $s : \mathsf{E}'$ dans Γ tel que les types E et E' soient incompatibles. Dans ces deux cas, le fait que l'environnement Γ_1 soit bien formé implique qu'au moins l'une des deux hypothèses est introduite par l'extension substitutive $[\cdot]_{t_1=t_2}$. On considère séparément les deux causes possibles de cette incompatibilité.
   - Si l'un des deux types est de la forme LOC ou $\mathsf{C}_@t$ alors que l'autre est de la forme $\mathsf{C}_@t$ ou $\tilde{t}^\star \mapsto t$, respectivement, on peut en conclure que Γ_1 contient les hypothèses $t_1 : \mathsf{E}$ et $t_2 : \mathsf{E}'$ ou vice versa. Par conséquent $\Gamma_2 \vdash (t_1 : \mathsf{E})\{^{S_2}/_{S_1}\}$ et $\Gamma_2 \vdash (t_2 : \mathsf{E}')\{^{S_2}/_{S_1}\}$. Cela implique immédiatement que Γ_2 contient des hypothèses $t_1\{^{S_2}/_{S_1}\} : \mathsf{E}_1$ et $t_2\{^{S_2}/_{S_1}\} : \mathsf{E}_2$ où E_1 a la même « forme » (localité, canal ou passeport) que E et E_2 que E'. On en déduit aisément que $[\Gamma_2]_{t_1\{^{S_2}/_{S_1}\}=t_2\{^{S_2}/_{S_1}\}} \not\vdash \mathbf{env}$.
   - Si les deux types sont de la même forme $\mathsf{C}_@t$ ou $\tilde{t}^\star \mapsto t$ mais les deux « t » qui apparaissent ainsi sont deux noms distincts, on distingue deux sous-cas. Si s est l'un des deux t_i une incompatibilité similaire est introduite dans $[\Gamma_2]_{t_1\{^{S_2}/_{S_1}\}=t_2\{^{S_2}/_{S_1}\}}$ car les deux « t » qui la provoquent sont des noms donc invariants par la substitution $\{^{S_2}/_{S_1}\}$. Sinon, on note les deux types incompatibles $\mathsf{E}(l_1)$ et $\mathsf{E}(l_2)$ où l_1 et l_2 sont les deux noms conflictuels. Cette incompatibilité ne peut être introduite que quand l'un des l_i est un t_i et l'autre ne l'est pas. Sans perte de généralité, on écrit $t_1 = l_1$. Ces deux hypothèses de $[\Gamma_1]_{t_1=t_2}$ sont engendrées par deux hypothèses $s : \mathsf{E}'(t_2)$ et $s : \mathsf{E}'(l_2)$ de Γ_1. Γ_2 contient alors des hypothèses $s\{^{S_2}/_{S_1}\} : \mathsf{E}''(t_2\{^{S_2}/_{S_1}\})$ et $s\{^{S_2}/_{S_1}\} : \mathsf{E}''(l_2\{^{S_2}/_{S_1}\}) = \mathsf{E}''(l_2)$. On considère alors les hypothèses qui en découlent dans $[\Gamma_2]_{t_1\{^{S_2}/_{S_1}\}=t_2\{^{S_2}/_{S_1}\}}$. Si $s\{^{S_2}/_{S_1}\}$ est l'un des $t_i\{^{S_2}/_{S_1}\}$, cet environnement sera mal formé, car on sait par ailleurs qu'il contient $t_i\{^{S_2}/_{S_1}\} : \text{LOC}$, car $\Gamma_2 \vdash \mathbf{env}$. Sinon il devra contenir des hypothèses $s\{^{S_2}/_{S_1}\} : \mathsf{E}'''(t_1\{^{S_2}/_{S_1}\}) = \mathsf{E}'''(l_1)$ et $s\{^{S_2}/_{S_1}\} : \mathsf{E}'''(l_2)$ soit deux hypothèses incompatibles, ce qui achève la preuve de ce cas.

2. Le type E mentionne des identifiants qui ne sont pas associés dans l'environnement Γ au type LOC. On suppose tout d'abord que E est de la forme $\mathsf{C}_{@}t$ avec t distinct des t_i. Alors Γ_1 doit être de la forme $\Gamma_1^1, s' : \mathsf{E}, \Gamma_1^2$ où $\left[\Gamma_1^1\right]_{t_1=t_2} = \Gamma$ et soit $s' = s$ soit $s' = t_i$ si $s = t_{3-i}$. La bonne formation de Γ_1 impose que $t : \text{LOC}$ est prouvable dans Γ_1^1 donc a fortiori également dans Γ.

   Un raisonnement similaire peut être tenu si t est l'un des t_i dans la mesure où une hypothèse $t_1 : \text{LOC}$ ou $t_2 : \text{LOC}$ dans Γ_1^1 entraînera que les deux sont prouvables dans Γ.

   Un dernier raisonnement similaire permet de conclure que le cas où le type E serait de la forme $\tilde{t}^\star \mapsto t$ est tout aussi impossible, ce qui achève la preuve de ce cas.

On a donc prouvé que la bonne formation de $[\Gamma_2]_{t_1\{^{S_2}/_{S_1}\}=t_2\{^{S_2}/_{S_1}\}}$ implique celle de $[\Gamma_1]_{t_1=t_2}$.

On suppose donc dans la suite que $[\Gamma_2]_{t_1\{^{S_2}/_{S_1}\}=t_2\{^{S_2}/_{S_1}\}}$ est bien formé et on considère $s : \mathsf{E}$, une hypothèse de $[\Gamma_1]_{t_1=t_2}$.

Si $(s : \mathsf{E}) \in \Gamma_1$, $\Gamma_2 \vdash (s : \mathsf{E})\{^{S_2}/_{S_1}\}$ donc en particulier $[\Gamma_2]_{t_1\{^{S_2}/_{S_1}\}=t_2\{^{S_2}/_{S_1}\}} \vdash (s : \mathsf{E})\{^{S_2}/_{S_1}\}$. Si $(s : \mathsf{E}) \notin \Gamma_1$, on sait alors qu'elle est introduite dans $[\Gamma_1]_{t_1=t_2}$ à partir d'un $s' : \mathsf{E}'$ de Γ_1. $s' : \mathsf{E}'$ doit donc mentionner t_1 ou t_2. Sans perte de généralité on suppose qu'il s'agit de t_1. On sait que t_1 ne peut être à la fois égal à s' et mentionné dans E' : la bonne formation de Γ_1 imposerait alors que t_1 soit de type LOC (pour être mentionné dans un autre type E') et que E' soit un type de canal ou de passeport donc incompatible avec LOC. On raisonne donc par cas.

- Si $t_1 = s'$, alors $s : \mathsf{E}$ doit être $t_2 : \mathsf{E}$; par hypothèse d'induction, on sait que $\Gamma_2 \vdash (t_1 : \mathsf{E})\{^{S_2}/_{S_1}\}$. Γ_2 contient donc des hypothèses $t_1\{^{S_2}/_{S_1}\} : \mathsf{E}^{(i)}$ telles que $\bigsqcap_i \mathsf{E}^{(i)} <: \mathsf{E}\{^{S_2}/_{S_1}\}$. Par conséquent $[\Gamma_2]_{t_1\{^{S_2}/_{S_1}\}=t_2\{^{S_2}/_{S_1}\}}$ contient $t_2\{^{S_2}/_{S_1}\} : \mathsf{E}^{(i)}$ pour tout i, ce qui implique $[\Gamma_2]_{t_1\{^{S_2}/_{S_1}\}=t_2\{^{S_2}/_{S_1}\}} \vdash t_2\{^{S_2}/_{S_1}\} : \mathsf{E}\{^{S_2}/_{S_1}\}$.
- Si $s' \notin \{t_1, t_2\}$ et $\mathsf{E}' = \mathsf{C}_{@}t_1$, alors $s = s'$ et, par hypothèse, on sait $\Gamma_2 \vdash s\{^{S_2}/_{S_1}\} : \mathsf{C}_{@}t_1\{^{S_2}/_{S_1}\}$ c'est-à-dire que l'environnement Γ_2 contient des hypothèses $s\{^{S_2}/_{S_1}\} : \mathsf{E}^{(i)}$ telles que $\bigsqcap_i \mathsf{E}^{(i)} <: \mathsf{C}_{@}t_1\{^{S_2}/_{S_1}\}$, ce qui implique que les $\mathsf{E}^{(i)}$ soient tous de la forme $\mathsf{C}^{(i)}{}_{@}t_1\{^{S_2}/_{S_1}\}$. L'hypothèse de bonne formation de $[\Gamma_2]_{t_1\{^{S_2}/_{S_1}\}=t_2\{^{S_2}/_{S_1}\}}$ impose que $s\{^{S_2}/_{S_1}\}$ ne soit pas égal à l'un des $t_i\{^{S_2}/_{S_1}\}$. Donc $[\Gamma_2]_{t_1\{^{S_2}/_{S_1}\}=t_2\{^{S_2}/_{S_1}\}}$ contient les hypothèses $s\{^{S_2}/_{S_1}\} : \mathsf{C}^{(i)}{}_{@}t_2\{^{S_2}/_{S_1}\}$, d'où l'on déduit $[\Gamma_2]_{t_1\{^{S_2}/_{S_1}\}=t_2\{^{S_2}/_{S_1}\}} \vdash s\{^{S_2}/_{S_1}\} : \mathsf{E}\{^{S_2}/_{S_1}\}$.
- Si $s' \notin \{t_1, t_2\}$ et $\mathsf{E} = \star \mapsto t_i$, le raisonnement est identique. Si $\mathsf{E} = \tilde{s} \mapsto s''$ avec $\{t_1, t_2\} \cap (\tilde{s} \cup \{s''\}) \neq \emptyset$ le raisonnement est similaire bien que plus complexe, notamment pour prendre en compte tous les cas de la définition de $[\cdot]_{t_1=t_2}$ sur les types de la forme $\cdot \mapsto \cdot$.

□

Lemme 4.9 (Substitution dans les processus)**.** *Soient Γ_1 et Γ_2 deux environnements, S_1 un motif pouvant contenir variables et variables de récursion et S_2 une valeur tels que pour tout $s : \mathsf{E}$ dans Γ_1, $\Gamma_2 \vdash s\{^{S_2}/_{S_1}\} : \mathsf{E}\{^{S_2}/_{S_1}\}$ soit prouvable. Alors pour tout jugement $\Gamma_1 \mid \Delta \vdash_t P$ prouvable, on peut prouver $\Gamma_2 \mid \Delta \vdash_{t\{^{S_2}/_{S_1}\}} P\{^{S_2}/_{S_1}\}$.*

Démonstration. On commence par remarquer que pour tout type clos T, si $s_0 : \mathsf{E}_0$ est une hypothèse de l'environnement $\Gamma_1;\langle X : \mathsf{T}\rangle_{@s}$ alors on peut prouver que $\Gamma_2;\langle X : \mathsf{T}\rangle_{@s}\{^{S_2}/_{S_1}\} \vdash (s_0 : \mathsf{E}_0)\{^{S_2}/_{S_1}\}$. En effet, si l'hypothèse est dans Γ_1, le résultat est immédiat. Par ailleurs, les hypothèses dans $\langle X : \mathsf{T}\rangle_{@s}$ sont de la forme $x_i : \mathsf{T}_i$ où le s rajouté par l'expansion est la seule variable de T_i qui puisse ne pas être invariante par la substitution $\{^{S_2}/_{S_1}\}$. Donc $(x_i : \mathsf{T}_i)\{^{S_2}/_{S_1}\}$ est soit $x_i : \mathsf{C}_{@s}\{^{S_2}/_{S_1}\}$ soit $x_i : \mathsf{T}_i$. Dans les deux cas de type T_i, cette hypothèse est dans $\Gamma_2;\langle X : \mathsf{T}\rangle_{@s}\{^{S_2}/_{S_1}\}$, ce qui achève de prouver que $\Gamma_2;\langle X : \mathsf{T}\rangle_{@s}\{^{S_2}/_{S_1}\} \vdash (s_0 : \mathsf{T}_0)\{^{S_2}/_{S_1}\}$.

Pour effectuer la preuve principale, on procède par induction sur la taille de la preuve de $\Gamma_1 \mid \Delta \vdash_s P$ et on raisonne sur la dernière règle utilisée dans cette preuve.

- (T-R) : $\Gamma_1 \mid \Delta \vdash_s u\,?\,(X : \mathsf{T})\,P$ implique à la fois que $\Gamma_1 \vdash_s u : \mathrm{R}\langle\mathsf{T}\rangle$ et $\Gamma_1;\langle X : \mathsf{T}\rangle_{@s} \mid \Delta \vdash_s P$. D'après la remarque faite précédemment, on peut appliquer l'hypothèse d'induction pour obtenir

$$\Gamma_2;\langle X : \mathsf{T}\rangle_{@s}\{^{S_2}/_{S_1}\} \mid \Delta \vdash_{s\{^{S_2}/_{S_1}\}} P\{^{S_2}/_{S_1}\}$$

  D'après le lemme 4.7, on peut aussi conclure $\Gamma_2 \vdash_{s\{^{S_2}/_{S_1}\}} (u : \mathrm{R}\langle\mathsf{T}\rangle)\{^{S_2}/_{S_1}\}$ c'est-à-dire $\Gamma_2 \vdash_{s\{^{S_2}/_{S_1}\}} u\{^{S_2}/_{S_1}\} : \mathrm{R}\langle\mathsf{T}\rangle$, selon les critères de formation des types transmissibles. Par (T-R), on peut maintenant déduire $\Gamma_2 \mid \Delta \vdash_{s\{^{S_2}/_{S_1}\}} (u\,?\,(X : \mathsf{T})\,P)\{^{S_2}/_{S_1}\}$.
- (T-IF) : la preuve de $\Gamma_1 \mid \Delta \vdash_s \mathsf{if}\ t_1 = t_2\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2$ repose donc sur $\Gamma_1 \mid \Delta \vdash_s P_2$ et $[\Gamma_1]_{t_1=t_2} \mid \Delta \vdash_s P_1$ dans le cas où $[\Gamma_1]_{t_1=t_2}$ est un environnement bien formé. Par hypothèse d'induction, on obtient que $\Gamma_2 \mid \Delta \vdash_{s\{^{S_2}/_{S_1}\}} P_2\{^{S_2}/_{S_1}\}$.
  D'après le lemme 4.8, si $[\Gamma_2]_{t_1\{^{S_2}/_{S_1}\}=t_2\{^{S_2}/_{S_1}\}} \vdash \mathbf{env}$ on doit avoir $[\Gamma_1]_{t_1=t_2} \vdash \mathbf{env}$ et on peut appliquer l'hypothèse d'induction avec les environnements $[\Gamma_1]_{t_1=t_2}$ et $[\Gamma_2]_{t_1\{^{S_2}/_{S_1}\}=t_2\{^{S_2}/_{S_1}\}}$ pour obtenir

$$[\Gamma_2]_{t_1\{^{S_2}/_{S_1}\}=t_2\{^{S_2}/_{S_1}\}} \mid \Delta \vdash_{s\{^{S_2}/_{S_1}\}} P_1\{^{S_2}/_{S_1}\}$$

  On peut donc en conclure que

$$\Gamma_2 \mid \Delta \vdash_{s\{^{S_2}/_{S_1}\}} (\mathsf{if}\ t_1 = t_2\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2)\{^{S_2}/_{S_1}\}$$

- Le raisonnement est similaire pour les autres règles.

□

Le dernier lemme de substitution nécessaire pour prouver la préservation du typage porte sur les appels récursifs.

Lemme 4.10 (Déroulement des appels récursifs)**.** *On suppose* $\Gamma \mid \Delta; Z : \mathsf{R} \vdash_s P$ *et* $\Gamma \mid \Delta \vdash_t (\mathit{rec}\ Z : \mathsf{R}\,(V)\,.\,Q)\,\langle X\rangle$. *Alors* $\Gamma \mid \Delta \vdash_s P[^{\mathsf{rec}\ Z:\mathsf{R}(X).\ Q}/_Z]$.

Démonstration. On opère cette preuve par induction sur la preuve de $\Gamma \mid Z : \mathsf{R}; \Delta \vdash_s P$.

- (T-HERE) : $\Gamma \mid \Delta; Z : \mathsf{R} \vdash_s \mathsf{here}\,(x)\,P$ implique $\Gamma \mid \Delta; Z : \mathsf{R} \vdash_s P\{^s/_x\}$. On veut prouver que $\Gamma \mid \Delta \vdash_s (\mathsf{here}\,(x)\,P)[^{\mathsf{rec}\ Z:\mathsf{R}(X).\ Q}/_Z]$. Cette substitution sans capture est définie dès lors que x n'est pas dans $\mathsf{fv}(\mathsf{rec}\ Z : \mathsf{R}\,(X)\,.\,Q) =$

$\mathsf{fv}(Q) \setminus \mathsf{fv}(X)$ et vaut simplement $\mathsf{here}\,(x)\,(P[^{\mathsf{rec}\ Z:\mathsf{R}(X).\ Q}/_Z])$. On se place dans ce cas, éventuellement par α-conversion.
Par ailleurs, l'hypothèse d'induction est $\Gamma \mid \Delta \vdash_s (P\{^s/_x\})[^{\mathsf{rec}\ Z:\mathsf{R}(X).\ Q}/_Z]$. D'après l'hypothèse émise précédemment, ce processus peut aussi être écrit $P[^{\mathsf{rec}\ Z:\mathsf{R}(X).\ Q}/_Z]\{^s/_x\}$. D'où l'on déduit par (T-HERE),

$$\Gamma \mid \Delta \vdash_s (\mathsf{here}\,(x)\,P)[^{\mathsf{rec}\ Z:\mathsf{R}(X).\ Q}/_Z]$$

- (T-CALL) : $\Gamma \mid \Delta; Z : \mathsf{R} \vdash_s Z\,\langle V_0\rangle$ implique $\Gamma \vdash_s (s, V_0) : \mathsf{R}$. Or $\Gamma \mid \Delta \vdash_t (\mathsf{rec}\ Z : \mathsf{R}\,(V)\,.\,P)\,\langle X\rangle$ repose sur le fait que $\Gamma;\langle (Z, X) : \mathsf{R}\rangle \mid \Delta; Z : \mathsf{R} \vdash_Z P$. ce qui permet de conclure $\Gamma \mid \Delta \vdash_s (\mathsf{rec}\ Z : \mathsf{R}\,(X)\,.\,P)\,\langle V_0\rangle$.
- (T-STOP) : le résultat est immédiat.
- (T-IF) : $\Gamma \mid \Delta; Z : \mathsf{R} \vdash_s \mathsf{if}\ s_1 = s_2\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2$ repose sur $\Gamma \mid \Delta; Z : \mathsf{R} \vdash_s P_2$ et, au cas où $[\Gamma]_{s_1=s_2} \vdash \mathbf{env}$, $[\Gamma]_{s_1=s_2} \mid \Delta; Z : \mathsf{R} \vdash_s P_1$. Par hypothèse d'induction, on obtient $\Gamma \mid \Delta \vdash_s P_2[^{\mathsf{rec}\ Z:\mathsf{R}(X).\ Q}/_Z]$. D'après la proposition 2.26 d'affaiblissement, on sait que $[\Gamma]_{s_1=s_2} \mid \Delta \vdash_t (\mathsf{rec}\ Z : \mathsf{R}\,(V)\,.\,Q)\,\langle X\rangle$ ce qui permet d'appliquer l'hypothèse d'induction pour obtenir $[\Gamma]_{s_1=s_2} \mid \Delta \vdash_s P_1[^{\mathsf{rec}\ Z:\mathsf{R}(X).\ Q}/_Z]$. On peut donc en conclure que $\Gamma \mid \Delta \vdash_s \mathsf{if}\ s_1 = s_2\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2$.
- Tous les autres cas se prouvent de façon similaire, à savoir en utilisant le lemme d'affaiblissement, les hypothèses d'induction et éventuellement des α-conversions.

□

La preuve de la préservation du typage repose enfin sur deux derniers lemmes portant sur l'extension substitutive. Comme promis au chapitre 2 (page 69), nous allons en effet prouver que, quel que soit le nom a, $[\Gamma]_{a=a}$ est un environnement tout simplement équivalent à Γ, ce qui permet de gérer la réduction (R-IF-V).

Lemme 4.11 (L'extension substitutive identitaire est incluse dans l'équivalence)**.** *Soient un environnement bien formé Γ et un nom a. L'environnement $[\Gamma]_{a=a}$ est alors bien formé et équivalent à Γ.*

Démonstration. Les deux preuves se font conjointement et procèdent, comme il se doit, par induction sur la preuve de $\Gamma \vdash \mathbf{env}$, en analysant chaque hypothèse de Γ. Si Γ est l'environnement vide, le résultat est immédiat. Pour l'environnement $\Gamma, s : \mathsf{E}$, on raisonne par cas sur la dernière hypothèse. Regardons uniquement un cas emblématique : quand $s \neq a$ et E est de la forme $a, \tilde{s} \mapsto a$. $[\Gamma, s : \mathsf{E}]_{a=a}$ sera alors l'environnement $[\Gamma]_{a=a}$ auquel cette dernière hypothèse est rajoutée deux fois sous la forme $s : a, a, \tilde{s} \mapsto a$, qui est exactement $s : \mathsf{E}$ car $a, a, \tilde{s}$ est évidemment le même ensemble que $a, \tilde{s}$. Il faut alors vérifier que $[\Gamma]_{a=a}\,(s)$ est compatible avec $a, a, \tilde{s} \mapsto a$. Or pour tout type E_0 de $[\Gamma]_{a=a}\,(s)$, on peut prouver dans l'environnement équivalent Γ que $\Gamma \vdash s : \mathsf{E}_0$. La bonne formation de $\Gamma, s : \mathsf{E}$ entraîne la compatibilité de $a, \tilde{s} \mapsto a$ avec $[\Gamma]_{a=a}\,(s)$. □

L'autre lemme porte sur l'utilisation de l'extension substitutive dans le typage de la construction here. On voudra en effet pouvoir conclure $\Gamma \mid \vdash_l P\{^l/_x\}$ de $[\Gamma]_{l=x} \mid \vdash_l P$.

Lemme 4.12. *Soient Γ un environnement bien formé, a un nom et x une variable tels que $x \notin \mathsf{dom}(\Gamma)$. Alors pour toute hypothèse $s : \mathsf{E}$ de $[\Gamma]_{a=x}$, il existe une hypothèse $(s : \mathsf{E})\{^a/_x\}$ dans Γ.*

Démonstration. Puisque la variable x n'apparaît pas dans l'environnement Γ, si l'hypothèse $s : \mathsf{E}$ ne mentionne pas la variable x, elle doit déjà être dans Γ donc le résultat est immédiat. Sinon, cette hypothèse a dû être introduite par l'extension substitutive. On raisonne donc sur le cas de la définition de cette extension (voir figure 2.11) qui correspond à l'hypothèse $s : \mathsf{E}$. Supposons par exemple qu'il s'agisse d'une hypothèse de la forme $s : x, a, \tilde{s} \mapsto x$. Cette hypothèse ne peut avoir été engendrée qu'à partir de $s : a, \tilde{s} \mapsto a$ c'est-à-dire exactement $(s : x, a, \tilde{s} \mapsto x)\{^{a}/_{x}\}$. Les autres cas sont similaires. □

Nous pouvons enfin prouver le théorème de préservation du typage dont l'énoncé reste inchangé par rapport à sa version dans le cadre du chapitre 2 (théorème 2.29).

Théorème 4.13 (Préservation du typage)**.** *Soit un système M bien typé dans l'environnement Γ et tel qu'il existe M' avec $M \longrightarrow M'$. Alors M' est bien typé dans Γ.*

Démonstration. On prouve ce résultat par induction sur la preuve de la réduction $M \longrightarrow M'$. On raisonne donc sur la dernière règle utilisée dans la preuve de cette réduction.

- (R-COMM) : l'hypothèse $\Gamma \vdash l[\![a\,!\,\langle V\rangle\, P_1]\!] \mid l[\![a\,?\,(X : \mathsf{T})\, P_2]\!]$ permet de déduire que $\Gamma \vdash_\iota a : \mathrm{W}\langle \mathsf{T}_0\rangle$, $\Gamma \vdash_\iota V : \mathsf{T}_0$, $\Gamma \mid \vdash_\iota P_1$, $\Gamma \vdash_\iota a : \mathrm{R}\langle \mathsf{T}\rangle$ et $\Gamma;\langle X : \mathsf{T}\rangle \,@l \mid \vdash_\iota P_2$. Par bonne formation de l'environnement Γ, les types $\mathrm{W}\langle \mathsf{T}_0\rangle$ et $\mathrm{R}\langle \mathsf{T}\rangle$ sont compatibles, T_0 est donc un sous-type de T. $\Gamma \vdash_\iota V : \mathsf{T}$ est donc prouvable.
  Soit $s_0 : \mathsf{T}_0$ une hypothèse de l'environnement $\Gamma;\langle X : \mathsf{T}\rangle \,@l$. Si cette hypothèse est dans la partie Γ de l'environnement, $\Gamma \vdash s_0 : \mathsf{T}_0$ est prouvable directement par (V-ID). Si elle est dans $\langle X : \mathsf{T}\rangle \,@l$, d'après le lemme 4.6, $\Gamma \vdash (s_0 : \mathsf{T}_0)\{^{V}/_{X}\}$. Le lemme 4.9 permet alors de conclure que $\Gamma;\langle X : \mathsf{T}\rangle \,@l \mid \vdash_\iota P_2$ implique $\Gamma \mid \vdash_\iota P_2\{^{V}/_{X}\}$, ce qui achève de prouver $\Gamma \vdash l[\![P_1]\!] \mid l[\![P_2\{^{V}/_{X}\}]\!]$.
- (R-GOTO) : l'hypothèse $\Gamma \vdash l[\![\mathsf{goto}_p\, k.\ P]\!]$ ne peut être prouvable qu'aux conditions $\Gamma \vdash p : l \mapsto k$ et $\Gamma \mid \vdash_k P$. La première condition, par bonne formation de l'environnement Γ implique que k soit une localité. On peut donc conclure de la seconde condition que $\Gamma \vdash k[\![P]\!]$.
- (R-IF-V) : quand $\Gamma \vdash l[\![\mathsf{if}\ a = a\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2]\!]$ est prouvable, on sait que si $[\Gamma]_{a=a}$ est un environnement bien formé, alors $[\Gamma]_{a=a} \mid \vdash_\iota P_1$. Le lemme 4.11 prouve justement la bonne formation de $[\Gamma]_{a=a}$ et son équivalence à l'environnement Γ. Par conséquent $\Gamma \mid \vdash_\iota P_1$, donc $\Gamma \vdash l[\![P_1]\!]$.
- (R-IF-F) : quand $\Gamma \vdash l[\![\mathsf{if}\ a_1 = a_2\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2]\!]$ on sait $\Gamma \mid \vdash_\iota P_2$ donc $\Gamma \vdash l[\![P_2]\!]$, ce qui achève la preuve dans ce cas.
- (R-NEWCHAN) : $\Gamma \vdash l[\![\mathsf{newchan}\, c : \mathsf{C}\, \mathsf{in}\, P]\!]$ repose sur $\Gamma; c : \mathsf{C}@l \mid \vdash_\iota P$ ce qui permet de conclure directement $\Gamma \vdash (\mathsf{new}\, c : \mathsf{C}@l)\, l[\![P]\!]$.
- (R-NEWLOC) et (R-NEWPASS) : le raisonnement est similaire au cas précédent.
- (R-SPLIT) et (R-RÉP) : le raisonnement est un simple dépliement de l'hypothèse que le processus est lui-même bien typé dans la localité donnée.
- (R-HERE) : $\Gamma \mid \vdash_\iota \mathsf{here}\,(x)\, P$ implique que $[\Gamma]_{l=x} \mid \vdash_\iota P$ avec x n'apparaissant pas dans Γ. Le lemme 4.12 fournit alors l'hypothèse nécessaire pour appliquer le lemme 4.9 qui permet de conclure que $\Gamma \mid \vdash_\iota P\{^{l}/_{x}\}$.

– (R-REC) : on veut prouver que $\Gamma \mid \vdash_l (\mathsf{rec}\ Z : \mathsf{R}(X).\ P)\langle V\rangle$ implique $\Gamma \mid \vdash_l P\{^V/_X\}[^{\mathsf{rec}\ Z:\mathsf{R}(X).\ P}/_Z]$. On peut tout d'abord déduire que $\Gamma \vdash_l (l, V) : \mathsf{R}$ et $\Gamma;\langle(Z, X) : \mathsf{R}\rangle \mid Z : \mathsf{R} \vdash_Z P$.
Soit $s_0 : \mathsf{T}_0$ une hypothèse de l'environnement $\Gamma;\langle(Z, X) : \mathsf{R}\rangle$. On en déduit que $\Gamma \vdash (s_0 : \mathsf{T}_0)\{^{(l,V)}/_{(Z,X)}\}$ que cette hypothèse soit dans Γ, auquel cas le résultat est direct, ou dans $\langle(Z, X) : \mathsf{R}\rangle$, auquel cas il est obtenu par le lemme 4.6. D'après le lemme 4.9, on peut en conclure $\Gamma \mid Z : \mathsf{R} \vdash_l P\{^{(l,V)}/_{(Z,X)}\}$. On sait par ailleurs que cette substitution se décompose en $\{^l/_Z\}$ et $\{^V/_X\}$, la première de ces deux substitutions laissant P invariant car un processus ne peut pas syntaxiquement contenir de variables de récursion en position d'identifiant. On sait donc $\Gamma \mid Z : \mathsf{R} \vdash_l P\{^V/_X\}$. D'après le lemme 4.10, on en déduit $\Gamma \mid \vdash_l P\{^V/_X\}[^{\mathsf{rec}\ Z:\mathsf{R}(V).\ P}/_Z]$, d'où le résultat est immédiat.
– (R-C-PAR) et (R-C-NEW) s'obtiennent par utilisation directe de l'hypothèse d'induction.
– (R-STRUCT) repose sur l'hypothèse d'induction et la proposition 4.5 de cohérence entre la congruence structurelle et le typage.

□

4.6 Équivalence observationnelle loyale

Délimitation de l'observable

Nous venons de décrire une variante du Dπ-calcul et de son typage dans laquelle les migrations de processus ne peuvent se faire sans autorisation. Ainsi, les systèmes ne se retrouvent plus sans défense face au contexte dans lequel ils sont placés. Puisque les contextes s'en retrouvent contrôlés, les observations que peuvent effectuer d'éventuels attaquants sont naturellement bridées. Par conséquent, il est pertinent de définir précisément les comportements observables d'un système dans ce cadre, et d'en extraire une notion d'équivalence qui sera une relation typée, exactement à l'image de l'équivalence observationnelle typée de la section 2.9.

Naturellement, nous suivrons la même approche d'équivalence typée, c'est-à-dire que les capacités d'observation sont caractérisées par un environnement de typage Ω représentant la connaissance de l'observateur. Comme précédemment, l'observateur devra ainsi disposer de la capacité de réception pour percevoir une émission sur un canal. Mais il devra aussi avoir le droit d'accéder à la localité dans laquelle cette émission a lieu. Il est donc fondamental de décrire précisément quelles sont ces localités. Revenons pour cela à l'intuition initiale : l'observé est une *boîte noire* dans les mains de l'observateur. La représentation naturelle de cette idée se traduit par le fait que l'observateur est ancré dans une *localité inédite*. Dans le cadre du chapitre 2, cela signifie qu'il peut effectuer ses observations dans toutes les localités dont il connaît le nom, les migrations n'étant pas entravées. Ici, il aura également besoin d'un passeport autorisant l'accès depuis sa localité : puisqu'elle est inédite pour l'observé qui délivre les passeports, il ne pourra s'agir que d'un *passeport universel*.

Formalisation de l'équivalence

On considérera donc qu'un observateur peut accéder directement à toutes les localités pour lesquelles il dispose d'un passeport universel, c'est-à-dire de type $\star \mapsto l$. On définit alors la congruence barbue qui découle de ce critère, en suivant la même démarche qu'au chapitre 2. Évidemment, les barbes typées ne pourront être observées que dans les localités ainsi accessibles.

Définition 4.14 (Barbes)**.** *On dit que la configuration $\Omega \rhd M$ exhibe une* barbe *sur a, ce que l'on note $\Omega \rhd M \Downarrow a$, si et seulement s'il existe une localité l et un passeport p tels que :*

- $\Omega \vdash p : \star \mapsto l$ *;*
- $\Omega \vdash a : \mathrm{R}\langle \mathsf{T} \rangle_{@l}$*, pour un type* T *quelconque ;*
- *et il existe P, M' et Φ tels que*

$$M \Longrightarrow\equiv (\mathit{new}\,\Phi)(M' \mid l[\![a\,!\,\langle V\rangle\, P]\!])$$

avec $a, l \notin \mathsf{dom}(\Phi)$.

Bien entendu, la notion de respect des barbes donnée à la définition 2.33 se transpose pour cette nouvelle catégorie de barbes. Regardons donc ce qui se passe concernant la contextualité en présence de contrôle des migrations. La définition 4.14 est conforme à l'intuition qu'une relation d'équivalence typée respectant les barbes $\mathcal{R}$ pourrait contenir

$$l : \mathrm{LOC}, c : \mathrm{RW}\langle\rangle_{@l} \vDash l[\![c\,!\,\langle\rangle]\!]\ \mathcal{R}\ \mathbf{0}$$

étant donné que l'observateur n'a pas la capacité d'aller vérifier si une barbe est exhibée dans la localité l puisqu'il ne possède aucun passeport pour y entrer. Cependant, si l'on conservait la définition 2.34 des relations contextuelles typées, il serait alors possible d'enrichir la connaissance de l'observateur de noms quelconques pour peu que ces noms soient inédits. Cela signifierait en particulier que $\mathcal{R}$ devrait aussi contenir

$$l : \mathrm{LOC}, c : \mathrm{RW}\langle\rangle_{@l}, p : \star \mapsto l \vDash l[\![c\,!\,\langle\rangle]\!]\ \mathcal{R}\ \mathbf{0}$$

puisque p est inédit. Cet exemple montre qu'il est impossible de conserver telle quelle la condition d'enrichissement de la connaissance de l'observateur sans créer un conflit avec le respect des barbes, car le premier des deux systèmes exhibe soudain une barbe après l'ajout du nom p à Ω. On imposera donc que ces noms inédits ne donnent pas indûment accès à des localités jusqu'à présent inobservables.

Avec un raisonnement très similaire à propos des contextes de la forme $[\cdot] \mid l[\![P]\!]$, on empêchera de conclure, à partir de :

$$l : \mathrm{LOC}, c : \mathrm{RW}\langle\rangle_{@l}, k : \mathrm{LOC}, d : \mathrm{RW}\langle\rangle_{@k}, p : \star \mapsto k \vDash l[\![c\,!\,\langle\rangle]\!]\ \mathcal{R}\ \mathbf{0}$$

que

$$\begin{aligned} l : \mathrm{LOC}, c : \mathrm{RW}\langle\rangle_{@l}, k : \mathrm{LOC}, d : \mathrm{RW}\langle\rangle_{@k}, p : \star \mapsto k \vDash \\ l[\![c\,!\,\langle\rangle]\!] \mid l[\![c\,?\,()\,\mathsf{goto}_p\, k.\ d\,!\,\langle\rangle]\!]\ \mathcal{R}\ \mathbf{0} \mid l[\![c\,?\,()\,\mathsf{goto}_p\, k.\ d\,!\,\langle\rangle]\!] \end{aligned}$$

car l'observateur s'autorise dans cet exemple à percevoir une barbe en déclenchant un processus dans une localité à laquelle il n'a pas accès. On imposera donc

que les contextes de la forme $[\cdot] \mid l[\![P]\!]$ ne soient utilisés que par des observateurs ayant déjà accès à la localité l.

Enfin considérons l'exemple suivant

$$l : \text{LOC}, p : \star \mapsto k, q : k \mapsto l \vDash M \mathcal{R} N$$

Cet exemple est intéressant parce que l'observateur connaît deux passeports : p lui donne un accès direct à la localité k, et q l'autorise à faire migrer des processus de k vers l. L'observateur dispose en somme d'un accès indirect à l. Afin d'obtenir la notion de contextualité autorisant le plus de contextes possibles[3], on voudrait par conséquent pouvoir aussi utiliser les contextes de la forme $[\cdot] \mid l[\![P]\!]$.

On formalise donc la notion d'accessibilité d'une localité pour un observateur Ω en définissant les chemins et l'ensemble des localités accessibles.

Définition 4.15 (Chemins)**.** *On dit qu'il existe un chemin entre les localités s_0 et s_1 dans l'environnement Γ, s'il existe des identifiants $u_1, \ldots, u_n$ et $t_0, \ldots, t_n$ tels que*

- $\Gamma \vdash u_i : t_{i-1} \mapsto t_i$;
- $t_0 = s_0$, $t_n = s_1$.

On note l'existence d'un tel chemin : $s_0 \mapsto_\Gamma^* s_1$.

On note $\star \mapsto_\Gamma^* s_1$ l'existence d'un chemin dont le premier passeport est universel.

Définition 4.16 (Localités accessibles)**.** *L'ensemble des* localités accessibles *pour un observateur connaissant Ω, noté $\mathcal{A}_\Omega$, est l'ensemble des localités l telles que $\star \mapsto_\Omega^* l$.*

La notion d'accessibilité permet de formaliser à quelles conditions l'observateur peut créer de nouveaux noms en respectant les contraintes du calcul. On pourra ainsi autoriser l'introduction de nouveaux passeports dès lors qu'ils ne permettent pas d'atteindre une localité jusqu'à présent inaccessible. Par conséquent, afin qu'il soit possible de rajouter de nouvelles localités et de les rendre accessibles, on devra rajouter simultanément une localité et le premier passeport qui y mène. De façon plus générale, on pourra introduire un nombre quelconque de noms simultanément, pourvu que cet ajout soit loyal.

Définition 4.17 (Extension loyale)**.** *Γ' est une* extension loyale *de Γ si :*

- $\Gamma; \Gamma' \vdash \mathbf{env}$;
- *pour toute hypothèse $u : \tilde{s}^\star \mapsto t$ de Γ', avec $t \in \mathsf{dom}(\Gamma)$, alors $t \in \mathcal{A}_\Gamma$;*
- *pour toute hypothèse $u : \mathsf{C}_@ s$ de Γ' avec $s \in \mathsf{dom}(\Gamma)$, alors $s \in \mathcal{A}_\Gamma$.*

Vérifions alors que cette définition atteint son objectif, à savoir contraindre les générations de noms effectuées par l'observateur à ne pas étendre ses pouvoirs d'observation.

Proposition 4.18 (Prévention des cambriolages)**.** *Si Γ' est une extension loyale de Γ, $\mathcal{A}_{\Gamma;\Gamma'} \cap \mathsf{dom}(\Gamma) = \mathcal{A}_\Gamma$.*

[3] Naturellement, le contexte $[\cdot] \mid l[\![P]\!]$ effectue exactement les mêmes observations que $[\cdot] \mid k[\![\mathsf{goto}_q\, l.\ P]\!]$. Cette extension des contextes englobés dans la contextualité pourra faciliter des preuves d'équivalence, mais ne modifiera en rien la notion de congruence barbue engendrée. Les extensions loyales suivent la même démarche : donner un maximum de libertés à l'observateur (contextes possibles et ajouts à l'environnement Ω) sans briser la protection que les passeports doivent apporter à l'observé.

Démonstration. L'inclusion $\mathcal{A}_\Gamma \subseteq \mathcal{A}_{\Gamma;\Gamma'} \cap \mathsf{dom}(\Gamma)$ découle directement de la définition des localités accessibles.

Sinon, soit t une localité de $\mathcal{A}_{\Gamma;\Gamma'} \cap \mathsf{dom}(\Gamma)$ et un chemin $u_1 : \star \mapsto t_1, \ldots, u_n : t_{n-1} \mapsto t$ qui prouve que t est accessible dans $\Gamma;\Gamma'$. Si tous les passeports u_i sont dans $\mathsf{dom}(\Gamma)$, t est immédiatement dans $\mathcal{A}_\Gamma$. Sinon on considère le dernier passeport u_i qui soit dans $\mathsf{dom}(\Gamma')$. On a alors un chemin $t_i \mapsto_\Gamma^* t$.

- Si $u_i = u_n$, la définition de la loyauté de Γ' impose que t soit accessible dans Γ.
- Si le passeport u_{i+1} est de type $\star \mapsto t_{i+1}$ alors $u_{i+1}, \ldots, u_n$ est un chemin qui prouve $t \in \mathcal{A}_\Gamma$.
- Sinon la localité t_i doit être mentionnée dans le type du passeport u_{i+1} donc être dans $\mathsf{dom}(\Gamma)$. La loyauté de Γ' impose alors que t_i soit accessible dans Γ et la composition de chemin prouvant $\star \mapsto_\Gamma^* t_i$ et $t_i \mapsto_\Gamma^* t$ permet de conclure $\star \mapsto_\Gamma^* t$.

□

On peut alors enfin définir la contextualité et en extraire une notion d'équivalence.

Définition 4.19 (Relation typée loyalement contextuelle)**.** *Une relation typée $\mathcal{R}$ entre systèmes est dite* loyalement contextuelle *si et seulement si les conditions suivantes sont vérifiées :*

- *Si $\Omega \vDash M \mathcal{R} N$, et Ω' est une extension loyale de Ω telle que $\mathsf{dom}(\Omega')$ ne contienne que des noms inédits alors $\Omega;\Omega' \vDash M \mathcal{R} N$.*
- *Si $\Omega \vDash M \mathcal{R} N$, $k \in \mathcal{A}_\Omega$ et $\Omega \vdash k[\![P]\!]$ alors $\Omega \vDash M \mid k[\![P]\!] \mathcal{R} N \mid k[\![P]\!]$.*
- *Si $\Omega; a : \mathsf{E} \vDash M \mathcal{R} N$ et les configurations $\Omega \rhd (\mathsf{new}\, a : \mathsf{E})\, M$ et $\Omega \rhd (\mathsf{new}\, a : \mathsf{E})\, N$ sont bien formées, alors $\Omega \vDash (\mathsf{new}\, a : \mathsf{E})\, M \mathcal{R} (\mathsf{new}\, a : \mathsf{E})\, N$.*

En somme, on a tous les outils pour la *congruence barbue loyale*, définie comme ses sœurs typée et non typée.

Définition 4.20 (Congruence typée barbue loyale fermée par réduction)**.** *On appelle* congruence typée barbue loyale fermée par réduction *la plus grande relation typée symétrique loyalement contextuelle, fermée par réduction et qui respecte les barbes typées. On la note $\cong^l$.*

4.7 Bisimilarité loyale

Poursuivant toujours la même démarche qu'au chapitre 2, nous allons chercher une définition alternative de la congruence typée barbue loyale fermée par réduction. En effet cette congruence possède la même faiblesse que $\cong^{tbr}$ concernant les preuves d'équivalence : pour prouver qu'une relation est loyalement contextuelle, il est nécessaire de passer en revue tous les contextes loyaux possibles. La solution avancée section 2.10 établit un système typé de transitions étiquetées de sorte que l'équivalence qu'il engendre, la bisimulation typée, coïncide avec $\cong^{tbr}$.

4.7.1 Système loyal de transitions étiquetées

Définissons donc un système de transitions étiquetées qui corresponde à la congruence loyale. Cette définition repose directement sur celle du STTE de la figure 2.13, seuls quelques cas sont modifiés. Ils sont donnés figure 4.9.

Fig. 4.9 Règles modifiées pour le système loyal de transitions étiquetées (voir figure 2.13)

$$\text{(TE-W)}\quad \frac{\begin{array}{l} l \in \mathcal{A}_\Omega \\ \Omega \vdash_l a : \mathrm{R}\langle \mathsf{T} \rangle \quad \text{avec } \mathsf{T} = \Omega^r(a) \end{array}}{\Omega \rhd l[\![a\,!\,\langle V \rangle\, P]\!] \xrightarrow{a!V} \Omega, \langle V : \mathsf{T} \rangle \circledast l \rhd l[\![P]\!]}$$

$$\text{(TE-R)}\quad \frac{\begin{array}{l} l \in \mathcal{A}_\Omega \\ \Omega \vdash_l a : \mathrm{W}\langle \mathsf{T}' \rangle \\ \Omega \vdash_l V : \mathsf{T}' \end{array}}{\Omega \rhd l[\![a\,?\,(X : \mathsf{T})\, P]\!] \xrightarrow{a?V} \Omega \rhd l[\![P\{^V/_X\}]\!]}$$

$$\text{(TE-GOTO)}\quad \Omega \rhd l[\![\mathsf{goto}_p\, k.\ P]\!] \xrightarrow{\tau} \Omega \rhd k[\![P]\!]$$

$$\text{(TE-NEWLOC)}\quad \begin{array}{l} \Omega \rhd l[\![\mathsf{newloc}\, k, (\tilde{c}), (\tilde{p}), (\tilde{q}) : \sum x : \mathrm{LOC}.\ \mathsf{T} \text{ with } P_k \text{ in } P]\!] \\ \qquad \xrightarrow{\tau} \Omega \rhd (\mathsf{new}\langle k, ((\tilde{c}), (\tilde{p}), (\tilde{q})) : \mathsf{T}\{^l/_x\} \rangle)\, k[\![P_k]\!] \mid l[\![P]\!] \end{array}$$

$$\text{(TE-NEWPASS)}\quad \Omega \rhd l[\![\mathsf{newpass}\, p \,\mathsf{from}\, \tilde{k} \,\mathsf{in}\, P]\!] \xrightarrow{\tau} \Omega \rhd (\mathsf{new}\, p : \tilde{k} \mapsto l)\, l[\![P]\!]$$

$$\text{(TE-OUV)}\quad \frac{\Omega \rhd M \xrightarrow{(\Phi)a!V} \Omega' \rhd M'}{\Omega \rhd (\mathsf{new}\, b : \mathsf{E})\, M \xrightarrow{(b:\mathsf{E};\Phi)a!V} \Omega' \rhd M'} \quad \begin{array}{l} b \neq a \\ b \in \mathsf{fn}(V) \cup \mathsf{fn}(\Phi) \end{array}$$

$$\text{(TE-AFF)}\quad \frac{\Omega; \Omega_e \rhd M \xrightarrow{(\Phi)a?V} \Omega' \rhd M'}{\Omega \rhd M \xrightarrow{(\Omega_e;\Phi)a?V} \Omega' \rhd M'} \quad \begin{array}{l} \mathsf{dom}(\Omega_e) \cap (\{a\} \cup \mathsf{fn}(M)) = \emptyset \\ \Omega_e \text{ est une extension loyale de } \Omega \end{array}$$

Comme ce fut le cas pour la sémantique, le nombre de modifications à opérer est faible. Les deux modifications majeures apportées par le système loyal de transitions étiquetées (abrégé en SLTE dans la suite) concernent les deux règles (TE-R) et (TE-W). En effet, dans ces deux règles, une contrainte supplémentaire est imposée : les localités où les préfixes sont réduits doivent être accessibles à l'observateur Ω pour que la transition puisse se faire. Cette modification est naturelle, puisqu'elle traduit simplement le fait que cette capacité supplémentaire est nécessaire pour que l'observateur puisse interagir avec le système.

Deux autres règles doivent subir des modifications importantes : (TE-AFF) et (TE-OUV). Pour la règle (TE-AFF), au lieu d'autoriser l'observateur à générer les noms uniquement l'un après l'autre, tout un environnement Ω' peut être introduit par une simple utilisation de la règle. Ce choix répond toujours au même problème, déjà exposé pour justifier la notion d'extension loyale (voir la définition 4.17) : il permet à l'observateur de générer simultanément une nouvelle localité et les passeports l'autorisant à y accéder. C'est aussi là qu'apparaît la loyauté du système de transitions étiquetées : on impose comme condition de bord le fait que Ω' soit une extension loyale de Ω pour que cette transition soit elle-même loyale.

Dans la règle (TE-OUV), seule la condition de bord est modifiée. Lors d'une émission, la portée du nom b doit être ouverte dès que b apparaît dans $\mathsf{fn}(V) \cup \mathsf{fn}(\Phi)$, c'est-à-dire soit dans la valeur soit dans les types d'autres noms dont la

portée est ouverte. Dans le chapitre 2, cette condition est simplifiée en $b \in \mathsf{fn}(V)$ car les types auxquels les valeurs sont transmises doivent être clos, ce qui impose que les localités mentionnées dans des types de la forme $\mathsf{C}@l$ soient aussi envoyées directement dans la valeur. En revanche, considérons un exemple où la version simplifiée est inadaptée en présence de types de passeports :

$$l : \text{LOC}, p_l : \star \mapsto l, a : \text{RW}\langle \sum x, y : \tilde{\text{LOC}}.\ x \mapsto y \rangle_@ l \triangleright \\ (\mathsf{new}\, k : \text{LOC})\,(\mathsf{new}\, k' : \text{LOC})\,(\mathsf{new}\, p : k, k' \mapsto l)\, l[\![a\,!\,\langle((k,l),(p))\rangle \mathsf{stop}]\!]$$

Dans cet exemple parfaitement *ad hoc*, le passeport p est envoyé au type $k \mapsto l$ qui est un sous-type du type $k, k' \mapsto l$ qui lui est associé. La seule transition qu'il est possible d'envisager pour ce système contient l'application suivante de la règle (TE-OUV) :

$$\frac{\Omega \triangleright (\mathsf{new}\, p : k, k' \mapsto l)\, l[\![a\,!\,\langle((k,l),(p))\rangle \mathsf{stop}]\!] \xrightarrow{(p:k,k'\mapsto l)a!((k,l),(p))} \Omega' \triangleright l[\![\mathsf{stop}]\!]}{\Omega \triangleright (\mathsf{new}\, k' : \text{LOC})\,(\mathsf{new}\, p : k, k' \mapsto l)\, l[\![a\,!\,\langle((k,l),(p))\rangle]\!] \xrightarrow{(k':\text{LOC},p:k,k'\mapsto l)a!((k,l),(p))} \Omega' \triangleright l[\![\mathsf{stop}]\!]}$$

où Ω désigne l'environnement $l : \text{LOC}, p_l : \star \mapsto l, a : \text{RW}\langle \sum x, y : \tilde{\text{LOC}}.\ x \mapsto y \rangle_@ l$, puisque le type de p dans l'étiquette mentionne le nom k'.

Le système loyal de transitions étiquetées définit une sémantique alternative du Dπ-calcul, via les transitions étiquetées par τ. Afin de montrer que la bisimulation reposant sur le SLTE que l'on introduira dans la suite puisse coïncider avec la congruence barbue loyale, la première propriété de cohérence à vérifier est le fait que la sémantique engendrée par le SLTE est égale à celle définie par la sémantique par réductions. Nous allons donc prouver que toute réduction silencieuse de l'une des sémantiques correspond à une réduction équivalente dans l'autre sémantique. Bien évidemment, l'obstacle essentiel à traiter est l'utilisation omniprésente de la congruence structurelle dans la sémantique par réductions. Cette preuve comportera ainsi plusieurs étapes :

- la remarque que les extensions loyales peuvent se composer, le résultat restant une extension loyale ; cette remarque rend la preuve suivante plus élégante (proposition 4.21) ;
- la vérification qu'une transition est possible pour un système si et seulement si elle est possible également pour tout système qui lui est structurellement congru (lemme 4.23) ;
- la vérification que toute réduction $M \longrightarrow M'$ de la sémantique par réductions implique l'existence d'une transition étiquetée par τ du système M vers un système équivalent à M' (lemme 4.24) ;
- enfin la vérification finale de la coïncidence des sémantiques, pour laquelle il ne reste plus que la réciproque du point précédent à vérifier (théorème 4.25).

Proposition 4.21 (Composition d'extensions loyales)**.** *Soient* Γ_1, Γ_2 *et* Γ_3 *trois environnements tels que* Γ_2 *et* Γ_3 *soient des extensions loyales de, respectivement,* Γ_1 *et* $\Gamma_1;\Gamma_2$. $\Gamma_2;\Gamma_3$ *est alors une extension loyale de* Γ_1.

Démonstration. $\Gamma_1;\Gamma_2;\Gamma_3 \vdash \mathbf{env}$ est automatique car Γ_3 est une extension loyale de $\Gamma_1;\Gamma_2$. Soit une hypothèse $u : \tilde{s}^\star \mapsto t$ de $\Gamma_2;\Gamma_3$, avec $t \in \mathsf{dom}(\Gamma_1)$. Si cette hypothèse est dans Γ_2, $t \in \mathcal{A}_{\Gamma_1}$ est immédiat car Γ_2 est une extension loyale de Γ_1. Sinon elle doit être dans Γ_3 donc $t \in \mathcal{A}_{\Gamma_1;\Gamma_2}$. Par le lemme 4.18, on en déduit $t \in \mathcal{A}_{\Gamma_1}$. Un raisonnement similaire pour les canaux localisés permet de conclure. □

Remarque 4.22 (Les réceptions peuvent finir par l'affaiblissement). Lorsqu'une configuration peut faire une transition $\xrightarrow{(\Phi)a?V}$, la preuve de cette transition peut être normalisée avec une seule application de la règle (TE-AFF) en position finale. En effet, on voit aisément que la règle (TE-AFF) commute avec les règles (TE-C-NEW) et (TE-C-PAR), car les noms de Φ sont inédits donc n'apparaissent pas, mêmes liés, dans la configuration. Qui plus est, la proposition 4.21 permet de conclure que plusieurs applications successives de (TE-AFF) peuvent être fusionnées.

Lemme 4.23 (Compatibilité entre la congruence structurelle et le SLTE)**.**

$$\text{Si}\quad \begin{array}{ccc} \Omega \rhd M_1 & \equiv_t & \Omega \rhd M^1 \\ {\scriptstyle\mu}\downarrow & & \\ \Omega' \rhd M_2 & & \end{array} \qquad \text{alors}\quad \begin{array}{ccc} \Omega \rhd M_1 & \equiv_t & \Omega \rhd M^1 \\ {\scriptstyle\mu}\downarrow & & \downarrow{\scriptstyle\bar{\mu}} \\ \Omega' \rhd M_2 & \equiv_t & \Omega' \rhd M^2 \end{array}$$

où $\bar{\mu}$ est

- *μ si $\mu = \tau$ ou $\mu = (\Phi)a?V$;*
- *$(\Phi)a!V$ si $\mu = (\Phi')a!V$ où Φ et Φ' sont identiques à une permutation des hypothèses près.*

Démonstration. Commençons par remarquer que deux systèmes structurellement équivalents sont bien typés dans les mêmes environnements d'après la proposition 4.5 ce qui permet de conclure que la bonne formation de la configuration $\Omega \rhd M_1$ implique celle de $\Omega \rhd M^1$. Remarquons aussi que toute configuration capable de faire une transition $\xrightarrow{(\Phi)a?V}$ est aussi capable de faire une transition $\xrightarrow{(\Phi')a?V}$ où Φ et Φ' sont identiques à une permutation des hypothèses près. Cela découle en effet de la remarque 4.22 : dans la preuve normalisée où la seule occurrence de (TE-AFF) est en position finale, Φ et Φ' peuvent être interchangés. Dans la suite de cette preuve, on prendra par conséquent implicitement en compte cette récriture possible de l'étiquette des transitions-réceptions, dans les cas où deux processus sont en train de communiquer et que l'étiquette de la transition du récepteur doit être adaptée pour l'émetteur après application de l'hypothèse d'induction.

La preuve s'effectue en effet par induction sur la preuve de l'équivalence $M_1 \equiv M^1$. Alors réflexivité et transitivité sont immédiates. La contextualité, c'est-à-dire quand $M_1 = \mathfrak{C}[M_1']$ et $M^1 = \mathfrak{C}[M^{1\prime}]$, se prouve en regardant le ou les préfixes de M_1 et M^1 qui agissent lors de la transition $\Omega \rhd M_1 \xrightarrow{\mu} \Omega' \rhd M_2$. S'ils sont uniquement dans le contexte $\mathfrak{C}$ ou uniquement dans M_1, le résultat est immédiat. Sinon, le contexte $\mathfrak{C}$ doit être de la forme $\mathfrak{C}'[M_{\mathfrak{C}} \mid \mathfrak{C}''[\cdot]]$, de sorte que la règle du SLTE qui prouve $\Omega \rhd M_{\mathfrak{C}} \mid \mathfrak{C}''[M_1] \xrightarrow{\tau} \Omega \rhd \cdots$ doit être (TE-COMM). Cette règle repose forcément sur $\Omega_{M_1} \rhd \mathfrak{C}''[M_1'] \xrightarrow{(\Phi)a!V} \cdots$ ou $\Omega_{M_1} \rhd \mathfrak{C}''[M_1'] \xrightarrow{(\Phi)a?V} \cdots$. Dans les deux cas, on peut en extraire une preuve d'une transition de la forme $\Omega_{M_1} \rhd M_1' \xrightarrow{\mu'} \Omega'_{M_1} \rhd M_2'$ sur laquelle appliquer l'hypothèse d'induction, qui peut alors communiquer avec le contexte, quitte à permuter les hypothèses dans l'étiquette de la transition que le contexte effectue.

Enfin les différents cas donnés figure 1.3 et leur symétrique donnent le résultat par une application assez immédiate de l'hypothèse d'induction. Regardons quelques cas.

- Le cas $M_1 \mid M_2 \equiv M_2 \mid M_1$ nécessite simplement de constater la symétrie des règles (TE-C-PAR) et (TE-COMM).

- Le cas $M_1 \mid (\mathsf{new}\, a : \mathsf{E})\, M_2 \equiv (\mathsf{new}\, a : \mathsf{E})(M_1 \mid M_2)$ suppose une analyse par cas selon que M_1, M_2 ou les deux prennent part à la transition :
  - si M_1 agit seul, la première règle de la preuve, (TE-C-PAR) peut être remplacée par les deux règles (TE-C-NEW) puis (TE-C-PAR) ou vice versa ; l'application de (TE-C-NEW) est garantie par l'hypothèse $a \notin \mathsf{fn}(M_1)$ et la convention de Barendregt sur les variables liées ;
  - si M_2 agit seul, la règle (TE-C-PAR) commute simplement avec la règle (TE-C-NEW) ou la règle (TE-OUV), suivant les cas ;
  - si les deux communiquent et que le nom a prend part à la communication, la règle (TE-OUV) est remplacée par la règle (TE-C-NEW) ou vice versa ; le système obtenu dans les deux cas est le même ;
  - si les deux communiquent et que le nom a ne prend pas part à la communication, le premier système devient $(\mathsf{new}\, \Phi)\, M_1' \mid (\mathsf{new}\, a : \mathsf{E})\, M_2'$ tandis que le second devient $(\mathsf{new}\, a : \mathsf{E})\, (\mathsf{new}\, \Phi)\, M_1' \mid M_2'$; l'hypothèse $a \notin \mathsf{n}(\mu)$ et la convention de Barendregt permettent de conclure que ces deux systèmes sont structurellement congrus en appliquant le nombre nécessaire de commutations des portées pour transformer le second en $(\mathsf{new}\, \Phi)\, (\mathsf{new}\, a : \mathsf{E})\, M_1' \mid M_2'$ et en finissant grâce au fait que $a \notin \mathsf{fn}(M_1')$.
- Le cas $(\mathsf{new}\, a : \mathsf{E}_a)\, (\mathsf{new}\, b : \mathsf{E}_b)\, M' \equiv (\mathsf{new}\, b : \mathsf{E}_b)\, (\mathsf{new}\, a : \mathsf{E}_a)\, M'$ entraîne au plus une commutation des deux règles qui traitent $(\mathsf{new}\, a : \mathsf{E}_a)$ et $(\mathsf{new}\, b : \mathsf{E}_b)$. Cette commutation peut modifier l'ordre d'apparition des deux noms a et b dans l'environnement apparaissant dans la transition $\xrightarrow{\mu}$ si les portées des deux noms doivent être ouvertes. Cet échange de position est traité par le passage de μ à $\bar{\mu}$.

□

Lemme 4.24 (Les réductions sont des actions invisibles)**.** *Si $M \longrightarrow M'$ et $\Omega \rhd M$ est une configuration bien formée, alors il existe $M'' \equiv M'$ tel que $\Omega \rhd M \xrightarrow{\tau} \Omega \rhd M''$.*

Démonstration. Procédons par induction sur la preuve de $M \longrightarrow M'$. Mis à part (R-COMM) et (R-STRUCT) toutes les règles de la sémantique par réductions existent aussi directement dans le SLTE. Regardons donc ces deux règles séparément.

Supposons que la réduction $M \longrightarrow M'$ soit

$$l[\![a\,!\,\langle V\rangle\, P]\!] \mid l[\![a\,?\,(X : \mathsf{T})\, Q]\!] \longrightarrow l[\![P]\!] \mid l[\![Q\{^V/_X\}]\!]$$

La bonne formation de $\Omega \rhd M$ entraîne l'existence de Γ tel que $\Gamma \vdash M$. On peut en déduire les jugements suivants : $\Gamma \vdash_\iota a : \mathrm{W}\langle \mathsf{T}'\rangle$, $\Gamma \vdash_\iota V : \mathsf{T}'$, et $\Gamma \vdash_\iota a : \mathrm{R}\langle \mathsf{T}''\rangle$. Puisque Γ dispose de la capacité de réception sur le canal a, on peut même prouver $\Gamma \vdash_\iota a : \mathrm{R}\langle \Gamma^r(a)\rangle$. Par conséquent

$$\Gamma \rhd l[\![a\,!\,\langle V\rangle\, P]\!] \xrightarrow{a!V} \Gamma, \langle V : \Gamma^r(a)\rangle \circledcirc l \rhd l[\![P]\!]$$

et

$$\Gamma \rhd l[\![a\,?\,(X : \mathsf{T})\, Q]\!] \xrightarrow{a?V} \Gamma \rhd l[\![Q\{^V/_X\}]\!]$$

ce qui achève de prouver

$$\Omega \rhd l[\![a\,!\,\langle V\rangle\, P]\!] \mid l[\![a\,?\,(X : \mathsf{T})\, Q]\!] \xrightarrow{\tau} \Omega \rhd l[\![P]\!] \mid l[\![Q\{^V/_X\}]\!]$$

Par conséquent si la réduction $M \longrightarrow M'$ n'utilise pas (R-STRUCT), on peut en obtenir directement une preuve que $\Omega \rhd M \xrightarrow{\tau} \Omega \rhd M'$.

Enfin l'hypothèse d'induction et le lemme 4.23 permettent de conclure directement dans le cas de la règle (R-STRUCT). □

Théorème 4.25 (Coïncidence des sémantiques)**.** *Les réductions du système loyal de transitions étiquetées et de la sémantique par réductions coïncident au sens des deux propriétés suivantes :*

- $\Omega \rhd M \xrightarrow{\tau} \Omega \rhd M'$ *implique que* $M \longrightarrow M'$ *;*
- $M \longrightarrow M'$ *implique qu'il existe* $M'' \equiv M'$ *avec* $\Omega \rhd M \xrightarrow{\tau} \Omega \rhd M''$.

Démonstration. La seconde partie de ce théorème est prouvée par le lemme 4.24.

La première se décompose en deux étapes :

1. prouver que $\Omega \rhd M \xrightarrow{(\Phi)a!V} \Omega' \rhd M'$ implique que M soit structurellement congru à un système de la forme $(\mathsf{new}\,\Phi)\,(\mathsf{new}\,\Phi')(l[\![a\,!\,\langle V\rangle\,P]\!] \mid N)$ avec $M' \equiv (\mathsf{new}\,\Phi')\,l[\![P]\!] \mid N$ et que $\Omega \rhd M \xrightarrow{(\Phi)a?V} \Omega' \rhd M'$ implique que M soit structurellement congru à un système de la forme
$$(\mathsf{new}\,\Phi')\,l[\![a\,?\,(X : \mathsf{T})\,Q]\!] \mid N$$
avec $M' \equiv (\mathsf{new}\,\Phi')\,l[\![Q\{^V/_X\}]\!] \mid N$;
2. prouver effectivement que le résultat tient pour τ.

Le premier point se montre par une simple induction sur la preuve de $\Omega \rhd M \xrightarrow{(\Phi)a!V} \Omega' \rhd M'$. Si la preuve se compose uniquement de la règle (TE-W), le résultat est immédiat. Si elle se termine par (TE-C-PAR) ou (TE-C-NEW), l'hypothèse d'induction couplée avec des extensions des portées des noms liés dans Φ et éventuellement dans Φ' permettent de conclure. Enfin si elle se termine par (TE-OUV), l'hypothèse d'induction achève directement la preuve. Le raisonnement est similaire pour $\xrightarrow{(\Phi)a?V}$.

Le second point se montre aussi par induction sur la preuve de $\Omega \rhd M \xrightarrow{\tau} \Omega \rhd M'$. Si la dernière règle est (TE-COMM), M se décompose en M_1 et M_2 avec :

- $\Omega_1 \rhd M_1 \xrightarrow{(\Phi)a!V} \Omega'_1 \rhd M'_1$
- $\Omega_2 \rhd M_2 \xrightarrow{(\Phi)a?V} \Omega'_2 \rhd M'_2$

Le premier point permet de conclure :

- $M_1 \equiv (\mathsf{new}\,\Phi)\,(\mathsf{new}\,\Phi')(l[\![a\,!\,\langle V\rangle\,P]\!] \mid N_1)$ avec $M'_1 \equiv (\mathsf{new}\,\Phi')\,l[\![P]\!] \mid N_1$;
- $M_2 \equiv (\mathsf{new}\,\Phi'')\,l[\![a\,?\,(X : \mathsf{T})\,Q]\!] \mid N_2$ avec $M'_2 \equiv (\mathsf{new}\,\Phi'')\,l[\![Q\{^V/_X\}]\!] \mid N_2$.

Par conséquent
$$M \equiv (\mathsf{new}\,\Phi)\,(\mathsf{new}\,\Phi')\,(\mathsf{new}\,\Phi'')(l[\![a\,!\,\langle V\rangle\,P]\!] \mid l[\![a\,?\,(X : \mathsf{T})\,Q]\!] \mid N_1 \mid N_2)$$
donc
$$M \longrightarrow (\mathsf{new}\,\Phi)\,(\mathsf{new}\,\Phi')\,(\mathsf{new}\,\Phi'')(l[\![P]\!] \mid l[\![Q\{^V/_X\}]\!] \mid N_1 \mid N_2) \equiv (\mathsf{new}\,\Phi)\,M'_1 \mid M'_2$$
ce qui achève de prouver $M \longrightarrow M'$ dans le cas où la dernière règle est (TE-COMM). Toutes les autres règles correspondent directement à une règle de la sémantique par réduction (en décomposant éventuellement (TE-C-PAR) en (R-C-PAR) et (R-STRUCT) pour obtenir la symétrie). □

Maintenant que l'égalité des sémantiques est établie, remarquons une dernière propriété qui garantit que le SLTE se conforme aux hypothèses que l'on a émise sur les systèmes : si les étiquettes des réceptions n'introduisent que des noms, la clôture des systèmes est préservée par toute transition du SLTE.

4.7.2 Équivalence engendrée par le système de transitions

Revenons sur l'objectif initial de la définition du système loyal de transitions étiquetées : donner une définition alternative de l'équivalence $\cong^l$. Toujours suivant la même démarche qu'au chapitre 2, cela passe par la définition des bisimulations engendrées par le SLTE. Malheureusement, la modification de la règle (TE-OUV) rend aussi nécessaire une modification de la définition 2.39 des actions que l'on extrait des transitions. La définition 2.39 engendre, à partir d'une transition

$$\Omega \rhd M \xrightarrow{(\Phi)a!V} \Omega' \rhd M'$$

l'action

$$\Omega \rhd M \xrightarrow{(\tilde{b})a!V} \Omega' \rhd M'$$

dès que $\tilde{b} = \mathsf{dom}(\Phi)$. Cette définition ne peut plus coïncider avec ce qui est réellement observable en présence de types de passeports. Le problème se situe au niveau des noms dont la portée doit être ouverte lorsqu'un processus émet un message. Le changement de conditions de bord sur la règle (TE-OUV) a déjà été évoqué plus haut. Prenons un exemple qui illustre la nécessaire divergence sur les actions. Si la connaissance de l'observateur est

$$\Omega = l : \textsc{loc}, k : \textsc{loc}, c : \textsc{rw}\langle \sum x, y : \textsc{l}\vec{\textsc{o}}\textsc{c}.\ x \mapsto y \rangle_{@} l, d : \textsc{rw}\langle \textsc{loc} \rangle_{@} l$$

alors les deux systèmes

$$l[\![\mathsf{newloc}\, l' : \textsc{loc}\ \mathsf{with\ stop\ in\ newpass}\, p\ \mathsf{from}\, k, l'\ \mathsf{in}\, c\,!\,\langle (k,l),(p)\rangle\, d\,!\,\langle l'\rangle]\!]$$

et

$$l[\![\mathsf{newloc}\, l' : \textsc{loc}\ \mathsf{with\ stop\ in\ newpass}\, p\ \mathsf{from}\, k\ \mathsf{in}\, c\,!\,\langle (k,l),(p)\rangle\, d\,!\,\langle l'\rangle]\!]$$

devraient intuitivement être indistinguables : le nouveau passeport p qui est généré par le processus est dans les deux cas transmis au même type $k \mapsto l$. Par conséquent, dans les deux cas, le nom l' n'est appris que lors de la communication sur le canal d. Cependant, en utilisant la définition 2.39 des actions, le premier des deux systèmes se réduirait de la façon suivante :

$$\begin{array}{rl} \xrightarrow{\tau}{}^2 & \Omega \rhd (\mathsf{new}\, l' : \textsc{loc})\,(\mathsf{new}\, p : k, l' \mapsto l)\, l[\![c\,!\,\langle (k,l),(p)\rangle\, d\,!\,\langle l'\rangle]\!] \\ \xrightarrow{(l',p)c!((k,l),(p))} & \Omega, p : k \mapsto l \rhd l[\![d\,!\,\langle l'\rangle]\!] \\ \xrightarrow{d!l'} & \Omega, p : k \mapsto l, l' : \textsc{loc} \rhd l[\![\mathsf{stop}]\!] \end{array}$$

c'est-à-dire que le nom l' est révélé en même temps que p. L'ouverture de la portée de l' est imposée par (TE-OUV) parce que celle de p est ouverte et que le type de p mentionne le nom l'. Qui plus est, puisque le nom l' n'est pas mentionné dans la valeur qui est transmise sur le canal c, l'environnement Ω ne l'apprend, pour ainsi dire, qu'au cours de la dernière action, sur le canal d. Par conséquent la dernière action qu'effectue le système, $d!l'$, ne lie pas le nom l' puisque le nom n'est pas lié dans la transition que le SLTE permet de prouver pour la configuration $\Omega, p : k \mapsto l \rhd l[\![d\,!\,\langle l'\rangle]\!]$. En revanche, le second système se réduirait de la façon suivante :

$$\begin{array}{rl} \xrightarrow{\tau}{}^2 & \Omega \rhd (\mathsf{new}\, l' : \textsc{loc})\,(\mathsf{new}\, p : k \mapsto l)\, l[\![c\,!\,\langle (k,l),(p)\rangle\, d\,!\,\langle l'\rangle]\!] \\ \xrightarrow{(p)c!((k,l),(p))} & \Omega, p : k \mapsto l \rhd (\mathsf{new}\, l' : \textsc{loc})\, l[\![d\,!\,\langle l'\rangle]\!] \\ \xrightarrow{(l')d!l'} & \Omega, p : k \mapsto l, l' : \textsc{loc} \rhd l[\![\mathsf{stop}]\!] \end{array}$$

Ces deux exemples illustrent le fait que l'ouverture de la portée d'un nom ne signifie absolument pas que l'observateur ait eu l'occasion d'apprendre ce nom. En effet, en poursuivant l'intuition des équivalences observationnelles, le contexte qu'un observateur pourrait utiliser pour interagir avec le système serait totalement incapable de distinguer les portées qui doivent être étendues pour que la communication sur le canal c puisse avoir lieu :

$$l[\![O]\!] \,|\, (\mathsf{new}\, l' : \textsc{loc})\, (\mathsf{new}\, p : k \mapsto l)\, l[\![c\,!\,\langle (k,l),(p)\rangle\, d\,!\,\langle l'\rangle]\!]$$

est bien entendu structurellement congru à

$$(\mathsf{new}\, l' : \textsc{loc})(l[\![O]\!] \,|\, (\mathsf{new}\, p : k \mapsto l)\, l[\![c\,!\,\langle (k,l),(p)\rangle\, d\,!\,\langle l'\rangle]\!])$$

dans la mesure où le nom l' est inconnu de l'observateur O. La seule différence entre les deux systèmes de l'exemple est uniquement que, au cours des réductions, la portée de l' englobera toujours celle de p dans le premier système alors que ces deux portées sont indépendantes dans le second. Mais cette dépendance est purement syntaxique : la structure du Dπ-calcul impose que les portées soient lexicales. La sémantique lève l'ambiguïté : tant que le nom l' n'est pas communiqué explicitement au processus O dans une valeur, celui-ci ne pourra pas le « connaître », c'est-à-dire que l' ne pourra pas apparaître syntaxiquement dans ce processus. La différence est donc rigoureusement invisible pour la congruence loyale $\cong^l$ et devra aussi l'être pour la bisimilarité afin que ces équivalences s'avèrent égales.

Afin de résoudre ce problème, on définit les actions élémentaires non plus sur de simples configurations de la forme $\Omega \rhd M$ mais sur des configurations annotées par un jeu de noms $\tilde{a}$ encore inconnus pour l'observateur, notées $\Omega \rhd_{\tilde{a}} M$.

Définition 4.26 (Actions)**.** *On dit que la configuration $\Omega \rhd_{\tilde{a}} M$ peut accomplir l'action μ et devient $\Omega' \rhd_{\tilde{a}'} M'$, et on le note $\Omega \rhd_{\tilde{a}} M \xrightarrow{\mu} \Omega' \rhd_{\tilde{a}'} M'$, si :*

- *on peut prouver $\Omega \rhd M \xrightarrow{\mu} \Omega' \rhd M'$ dans le système loyal de transitions étiquetées quand μ est de la forme τ ou $(\Phi)a?V$ et $\tilde{a} = \tilde{a}'$;*
- *on peut prouver $\Omega \rhd M \xrightarrow{(\Phi)a!V} \Omega' \rhd M'$ dans le système loyal de transitions étiquetées quand μ est de la forme $(\tilde{b})a!V$ où $\tilde{b} = \mathsf{fn}(V) \cap (\mathsf{dom}(\Phi) \cup \tilde{a})$ et $\tilde{a}' = (\mathsf{dom}(\Phi) \cup \tilde{a}) \setminus \mathsf{fn}(V)$.*

Bien qu'il soit indispensable d'annoter les configurations par un ensemble de noms qui ne sont pas encore connus pour définir les actions, ces annotations ne sont pas nécessaires dans la formulation du SLTE. L'indication $\tilde{a}$ ne sert en effet qu'à lier des noms, de sorte que, suivant la convention de Barendregt, aucun conflit ne soit possible avec les noms connus ou introduits par l'observateur. Elle ne change cependant en rien la dérivation d'une transition. C'est pourquoi les deux sortes de configuration $\Omega \rhd M$ et $\Omega \rhd_{\tilde{a}} M$ cohabiteront dans la suite de ce chapitre : les premières effectueront des transitions et les secondes des actions. L'ambiguïté qui existait entre transitions et actions au chapitre 2 sera ainsi levée par la force des choses.

Avec cette définition des actions, les suites d'actions des deux systèmes de l'exemple précédent deviennent :

$$\begin{array}{rl}
\Omega \rhd_{\tilde{a}} l[\![\mathsf{newloc}\, l' : \textsc{loc}\, \mathsf{with\ stop\ in\ newpass}\, p\, \mathsf{from}\, k, l'\, \mathsf{in}\, c\,!\,\langle (k,l),(p)\rangle\, d\,!\,\langle l'\rangle]\!] & \\
\xrightarrow{\tau}{}^2 & \Omega \rhd_{\tilde{a}} (\mathsf{new}\, l' : \textsc{loc})\, (\mathsf{new}\, p : k, l' \mapsto l)\, l[\![c\,!\,\langle (k,l),(p)\rangle\, d\,!\,\langle l'\rangle]\!] \\
\xrightarrow{(p)c!((k,l),(p))} & \Omega, p : k \mapsto l \rhd_{\tilde{a},l'} l[\![d\,!\,\langle l'\rangle]\!] \\
\xrightarrow{(l')d!l'} & \Omega, p : k \mapsto l, l' : \textsc{loc} \rhd_{\tilde{a}} l[\![\mathsf{stop}]\!]
\end{array}$$

et

$$\begin{array}{rl} \Omega \rhd_{\tilde{a}} l[\![\mathsf{newloc}\, l' : \mathrm{LOC}\, \mathsf{with\ stop\ in\ newpass}\, p\, \mathsf{from}\, k\, \mathsf{in}\, c\,!\,\langle (k,l),(p)\rangle\, d\,!\,\langle l'\rangle]\!] \\ \xrightarrow{\tau}{}^{2} & \Omega \rhd_{\tilde{a}} (\mathsf{new}\, l' : \mathrm{LOC})\, (\mathsf{new}\, p : k \mapsto l)\, l[\![c\,!\,\langle (k,l),(p)\rangle\, d\,!\,\langle l'\rangle]\!] \\ \xrightarrow{(p)c!((k,l),(p))} & \Omega, p : k \mapsto l \rhd_{\tilde{a}} (\mathsf{new}\, l' : \mathrm{LOC})\, l[\![d\,!\,\langle l'\rangle]\!] \\ \xrightarrow{(l')d!l'} & \Omega, p : k \mapsto l, l' : \mathrm{LOC} \rhd_{\tilde{a}} l[\![\mathsf{stop}]\!] \end{array}$$

À partir de cette nouvelle définition des actions, on définit les actions faibles et les environnements résultants exactement de la même manière que précédemment (voir les définitions 2.41 et 2.40). Les relations typées sont quant à elles modifiées pour prendre en compte les annotations $\tilde{a}$.

Définition 4.27 (Relation typée annotée)**.** *On appellera* relation typée annotée *une relation $\mathcal{R}$ portant sur des quintuplets $(\Omega, \tilde{a}_M, M, \tilde{a}_N, N)$ telle que $\Omega \rhd_{\tilde{a}_M} M$ et $\Omega \rhd_{\tilde{a}_N} N$ soient des configurations annotées.*

On notera $\Omega \vDash M\ {}_{\tilde{a}_M}\mathcal{R}_{\tilde{a}_N}\ N$ l'appartenance du quintuplet $(\Omega, \tilde{a}_M, M, \tilde{a}_N, N)$ à la relation $\mathcal{R}$.

Les *bisimulations loyales* sont des relations typées annotées définies à partir des actions et actions faibles de façon très similaire aux bisimulations typées (définition 2.42).

Définition 4.28 (Bisimulation loyale)**.** *La relation typée annotée $\mathcal{R}$ est une* bisimulation loyale *si $\Omega \vDash M\ {}_{\tilde{a}_M}\mathcal{R}_{\tilde{a}_N}\ N$ implique :*

- *si $\Omega \rhd_{\tilde{a}_M} M \xrightarrow{\mu} \Omega$ $\mathsf{après}$ $\mu \rhd_{\tilde{a}'_M} M'$ alors il doit exister un système N' tel que $\Omega \rhd_{\tilde{a}_N} N \xRightarrow{\hat{\mu}} \Omega$ $\mathsf{après}$ $\mu \rhd_{\tilde{a}'_N} N'$ et Ω $\mathsf{après}$ $\mu \vDash M'\ {}_{\tilde{a}'_M}\mathcal{R}_{\tilde{a}'_N}\ N'$;*
- *symétriquement, si $\Omega \rhd_{\tilde{a}_N} N \xrightarrow{\mu} \Omega$ $\mathsf{après}$ $\mu \rhd_{\tilde{a}'_N} N'$ alors il doit exister un système M' tel que $\Omega \rhd_{\tilde{a}_M} M \xRightarrow{\hat{\mu}} \Omega$ $\mathsf{après}$ $\mu \rhd_{\tilde{a}'_M} M'$ et Ω $\mathsf{après}$ $\mu \vDash M'\ {}_{\tilde{a}'_M}\mathcal{R}_{\tilde{a}'_N}\ N'$.*

On peut enfin définir la bisimilarité loyale.

Définition 4.29 (Bisimilarité loyale)**.** *On appelle* bisimilarité loyale *la plus grande bisimulation loyale. On la note $\approx^{al}$.*

4.7.3 Congruence loyale annotée

Rappelons-nous que l'objectif de la définition de la bisimulation loyale était d'obtenir une définition alternative et plus manipulable de la congruence barbue loyale. Or nous disposons maintenant de deux notions d'équivalence incomparables : la congruence barbue loyale est une relation sur les configurations alors que la bisimilarité loyale porte sur les configurations annotées. Pour résoudre cette difficulté, définissons la congruence barbue loyale annotée :

Définition 4.30 (Relation typée annotée loyalement contextuelle)**.** *Une relation typée annotée $\mathcal{R}$ entre systèmes est dite* loyalement contextuelle *si et seulement si les conditions suivantes sont vérifiées.*

- *Si $\Omega \vDash M\ {}_{\tilde{a}_M}\mathcal{R}_{\tilde{a}_N}\ N$, et Ω' est une extension loyale de Ω telle que $\mathsf{dom}(\Omega')$ ne contienne que des noms inédits alors $\Omega; \Omega' \vDash M\ {}_{\tilde{a}_M}\mathcal{R}_{\tilde{a}_N}\ N$.*
- *Si $\Omega \vDash M\ {}_{\tilde{a}_M}\mathcal{R}_{\tilde{a}_N}\ N$, $k \in \mathcal{A}_\Omega$, $\Omega \vdash k[\![P]\!]$ et $\mathsf{fn}(P) \cap (\tilde{a}_M \cup \tilde{a}_N) = \emptyset$ alors $\Omega \vDash M \mid k[\![P]\!]\ {}_{\tilde{a}_M}\mathcal{R}_{\tilde{a}_N}\ N \mid k[\![P]\!]$.*

– *Si* $\Omega; b : \mathsf{E} \vDash M \ {}_{\tilde{a}_M}\mathcal{R}_{\tilde{a}_N} \ N$ *et que les configurations annotées* $\Omega \rhd_{\tilde{a}_M} (\mathit{new}\, b : \mathsf{E})\, M$ *et* $\Omega \rhd_{\tilde{a}_N} (\mathit{new}\, b : \mathsf{E})\, N$ *sont bien formées, alors*

$$\Omega \vDash (\mathit{new}\, b : \mathsf{E})\, M \ {}_{\tilde{a}_M}\mathcal{R}_{\tilde{a}_N} \ (\mathit{new}\, b : \mathsf{E})\, N$$

Cette définition de la contextualité est très similaire à la définition 4.19. La principale différence entre les deux est cachée dans le fait que les noms libres des contextes parallèles $k[\![P]\!]$ ne doivent pas entrer en conflit avec les noms liés des annotations. En effet, l'ajout du système $k[\![P]\!]$ en parallèle le place intuitivement dans la portée de ces noms liés, qu'il ne peut qu'ignorer. Bien évidemment, cette contextualité est définie pour obtenir une congruence annotée :

Définition 4.31 (Congruence typée annotée barbue loyale fermée par réduction)**.** *On appelle* congruence typée annotée barbue loyale fermée par réduction *la plus grande relation typée symétrique loyalement contextuelle, fermée par réduction et qui respecte les barbes typées. On la note* $\cong^{al}$*.*

Cette nouvelle congruence fera le lien entre la bisimulation loyale et la congruence barbue loyale.

Proposition 4.32 (Adéquation entre la congruence barbue loyale et sa variante annotée)**.** $\Omega \vDash M \ {}_{\emptyset}\cong^{al}_{\emptyset} \ N$ *si et seulement si* $\Omega \vDash M \cong^{l} N$*.*

Démonstration. L'égalité de ces deux équivalences sur leur fragment commun de définition découle du fait qu'aucune des conditions de définition de $\cong^{al}$ ne modifie les annotations. Par conséquent, en définissant la relation $\mathcal{R}$ ainsi :

$$\Omega \vDash M \ \mathcal{R} \ N \quad \text{si } \Omega \vDash M \ {}_{\emptyset}\cong^{al}_{\emptyset} \ N$$

$\mathcal{R}$ est naturellement une relation typée symétrique loyalement contextuelle, fermée par réduction et qui respecte les barbes typées, par définition de $\cong^{al}$. Donc $\mathcal{R} \subseteq \cong^{l}$. On voit aussi que la relation $\mathcal{S}$ définie par :

$$\Omega \vDash M \ {}_{\emptyset}\mathcal{S}_{\emptyset} \ N \quad \text{si } \Omega \vDash M \cong^{l} N$$

est incluse dans $\cong^{al}$, car $\mathcal{S}$ est une relation typée symétrique loyalement contextuelle, fermée par réduction et qui respecte les barbes typées. □

4.7.4 Outils

Nous allons pouvoir nous attacher à prouver la coïncidence de la bisimulation loyale et de la congruence barbue loyale annotée pour finir cette section. Commençons cette investigation en prouvant que la bisimulation loyale est incluse dans la congruence loyale annotée. Puisque la congruence est la plus grande relation respectant un jeu d'hypothèses vérifions tour à tour ces hypothèses dans le cas de la bisimilarité. La plus complexe parmi les hypothèses à vérifier est certainement la contextualité loyale. Cette preuve est alors découpée entre les trois critères qui définissent cette caractéristique. Mais, avant de pouvoir prouver les trois propriétés correspondant aux trois critères de contextualité loyale, nous allons avoir besoin de développer un peu plus avant la théorie de la bisimulation loyale.

– Nous allons tout d'abord nous assurer que deux systèmes structurellement congrus sont bisimilaires (corollaire 4.33).

- Nous verrons ensuite au lemme 4.35 un cas où l'on peut déduire d'une transition d'une configuration annotée $\Omega_1 \rhd_{\tilde{a}_M} M$ qu'une autre configuration annotée $\Omega_2 \rhd_{\tilde{a}_M} M$ sera capable d'effectuer la même transition. On pourra en conclure que c'est le cas si Ω_2 est un sous-type de Ω_1 (corollaire 4.36) et que deux systèmes bisimilaires dans un environnement Ω le sont aussi dans tout environnement sur-type de Ω (corollaire 4.37).
- Enfin, nous approfondirons la commutation d'hypothèses dans les environnements (proposition 4.38) afin de pouvoir associer des formes normales aux environnements (proposition 4.39) dont une forme normale naturelle (définition 4.40).

La première propriété dont on aura besoin est un simple corollaire du lemme 4.23 de compatibilité entre la congruence structurelle et le SLTE.

Corollaire 4.33 (La congruence structurelle typée est une bisimulation loyale)**.** $\equiv_t \subseteq \approx^{al}$.

Démonstration. Le lemme 4.23 implique que, quelles que soient les configurations dans lesquelles ils sont placés, deux systèmes structurellement congrus peuvent effectuer les mêmes actions. En effet, la seule modification possible sur les étiquettes des transitions est une permutation sur l'ensemble des noms liés dans les émissions. Or l'action correspondant à ces émissions ne conserve de l'environnement mentionné dans l'étiquette que l'ensemble de ses noms qui apparaissent dans la valeur transmise. □

On utilisera par ailleurs la technique de preuve « bisimulation modulo congruence structurelle » pour prouver la bisimilarité de systèmes. Il s'agit là d'un cas tout à fait rudimentaire de la technique de preuve décrite au chapitre 3 de bisimulation modulo β-réduction et bisimilarité forte. En effet, la congruence structurelle est incluse dans la bisimilarité forte par le même argument que le corollaire 4.33.

Comme annoncé, voyons maintenant sous quelles conditions les transitions effectuées dans un environnement Ω peuvent aussi avoir lieu dans Ω'. L'environnement Ω qui représente les connaissances que l'observateur possède sur le système M qu'il observe bride l'ensemble des actions que la configuration annotée $\Omega \rhd_{\tilde{a}_M} M$ est apte à effectuer. Puisque Ω détermine entièrement si une action donnée est possible, on dira que l'environnement Ω active l'action μ s'il existe un système M tel que $\Omega \rhd_{\tilde{a}_M} M \xrightarrow{\mu} \Omega' \rhd_{\tilde{a}'_M} M'$. On peut même établir facilement les critères qui déterminent si une action donnée est activée ou non dans un environnement donné. En effet, on voit par une simple induction sur la preuve d'une transition $\xrightarrow{(\Phi)c!V}$ qu'elle n'est possible que quand Ω dispose de la capacité de réception pour le canal c et peut accéder à la localité l où le canal c est situé. Les conditions duales s'appliquent pour la réception. On définit donc l'activation des actions de la façon suivante :

Définition 4.34 (Activation d'une action)**.** *On dit que l'environnement* Ω active *l'action* μ *si :*

- $\Omega \vdash_l c : \mathrm{R}\langle \mathsf{T} \rangle$ *avec* $l \in \mathcal{A}_\Omega$ *quand* $\mu = (\tilde{b})c!V$ *;*
- $\Omega \vdash_l c : \mathrm{W}\langle \mathsf{T} \rangle$ *avec* $l \in \mathcal{A}_\Omega$*,* Φ *est une extension loyale de* Ω *et* $\Omega; \Phi \vdash_l V : \mathsf{T}$ *quand* $\mu = (\Phi)c?V$ *;*
- *ou* $\mu = \tau$*.*

Les conditions d'activation de l'action μ permettent alors de prouver que, si un système effectue une transition correspondant à μ dans un environnement donné, cette transition est possible dans tout environnement où μ est activée.

Lemme 4.35 (Renforcement des transitions)**.** *Si la transition* $\Omega \rhd M \xrightarrow{\mu} \Omega$ après $\mu \rhd M'$ *est prouvable, que l'action associée à* μ *est activée par l'environnement* Ω'*, et que* $\Omega' \rhd M$ *est bien formée, alors la transition* $\Omega' \rhd M \xrightarrow{\mu} \Omega'$ après $\mu \rhd M'$ *est également prouvable.*

Démonstration. Cette preuve fonctionne par induction sur la dérivation de la transition $\Omega \rhd M \xrightarrow{\mu} \Omega$ après $\mu \rhd M'$. Si la dernière règle de cette transition est un des axiomes ((TE-GOTO), (TE-IF-V), (TE-IF-F), (TE-NEWCHAN), (TE-NEWLOC), (TE-NEWPASS), (TE-SPLIT), (TE-HERE), (TE-REP) et (TE-REC)) le résultat est direct parce qu'aucune condition ne porte sur l'environnement de la configuration qui est en jeu dans la transition. De façon similaire, s'il s'agit de la règle (TE-COMM), étant donné que les conditions ne portent aucunement sur Ω, le résultat est direct.

Si la dernière règle est (TE-R), la transition doit être de la forme

$$\Omega \rhd l[\![c\,!\,\langle V\rangle\, P]\!] \xrightarrow{c!V} \Omega, \langle V : \Omega^r(c)\rangle \circledast l \rhd l[\![P]\!]$$

Puisque $c!V$ est activée dans Ω', la transition

$$\Omega' \rhd l[\![c\,!\,\langle V\rangle\, P]\!] \xrightarrow{c!V} \Omega', \langle V : \Omega'^r(c)\rangle \circledast l \rhd l[\![P]\!]$$

est également prouvable. La même remarque s'applique pour la règle (TE-W).

Les règles (TE-C-PAR), (TE-C-NEW) et (TE-OUV) se traitent simplement en utilisant immédiatement l'hypothèse d'induction.

S'il s'agit de (TE-AFF) et de la transition $(\Phi; \Phi')c?V$, l'activation de l'action dans Ω' permet de garantir que Φ est aussi une extension loyale de Ω'. Par ailleurs, les critères d'activation de l'action permettent de garantir que $(\Phi')c?V$ est activée dans $\Omega'; \Phi$, donc l'hypothèse d'induction permet de conclure. □

Ce lemme permet de voir facilement que les transitions effectuées dans un environnement sont toujours possibles dans les environnements sous-types.

Corollaire 4.36 (Affaiblissement des transitions)**.** *Si la transition* $\Omega \rhd M \xrightarrow{\mu} \Omega$ après $\mu \rhd M'$ *est prouvable et que l'environnement* Ω' *est un sous-type de* Ω *tel que* $\Omega' \rhd M$ *soit bien formée, alors la transition* $\Omega' \rhd M \xrightarrow{\mu} \Omega'$ après $\mu \rhd M'$ *est également prouvable.*

Démonstration. Il suffit de remarquer que toute action activée dans Ω est aussi activée dans tout environnement Ω' sous-type de Ω. Pour cela, vérifions que $\mathcal{A}_\Omega \subseteq \mathcal{A}_{\Omega'}$. $l \in \mathcal{A}_\Omega$ signifie qu'il existe des noms $p_1, \ldots, p_n$ de passeport, et $l_1, \ldots, l_{n-1}$ de localité tels que $\Omega \vdash p_1 : \star \mapsto l_1$, pour tout $1 < i < n$, $\Omega \vdash p_i : l_{i-1} \mapsto l_i$ et $\Omega \vdash p_n : l_{n-1} \mapsto l$. La proposition 4.3 permet de prouver ces jugements dans l'environnement sous-type Ω', et donc de prouver que l est aussi accessible dans Ω'.

Vérifions l'autre condition : supposons que Φ soit une extension loyale de l'environnement Ω. Φ est alors aussi une extension loyale pour Ω'. En effet, on obtient $\Omega'; \Phi \vdash \mathbf{env}$ en raisonnant sur la taille de Φ. Les noms du domaine de Φ étant choisis inédits, on peut assurer qu'ils n'apparaissent pas dans Ω'. Supposons donc $\Omega'; \Phi'$ bien formé et Φ de la forme $\Phi', a : \mathsf{E}, \ldots$

- Si $\mathsf{E} = \text{LOC}$, comme a ne peut pas apparaître dans Ω', la bonne formation de $\Omega;\Phi$ impose que si $a \in (\Omega';\Phi')$ implique qu'il soit déjà associé au type LOC dans Φ'.
- Si $\mathsf{E} = \mathsf{C}_{@}s$, s doit être une localité dans Φ' si elle n'est pas dans le domaine de Ω ; si elle est dans $\mathsf{dom}(\Omega)$, l'hypothèse $s : \text{LOC}$ que Ω contient alors forcément implique que cette hypothèse apparaisse aussi dans Ω'. Par ailleurs, la compatibilité au sens large avec $(\Omega';\Phi')(a)$ se ramène à une compatibilité au sens large avec $\Phi'(a)$ car $a \notin \mathsf{dom}(\Omega')$ donc elle découle directement de la bonne formation de $\Omega;\Phi$.
- Le même raisonnement peut être tenu si $\mathsf{E} = \tilde{s}^\star \mapsto s$.

Considérons maintenant une hypothèse $p : \tilde{l}^\star \mapsto l$ de Φ telle que $l \in \mathsf{dom}(\Omega')$. Puisque $\mathsf{dom}(\Omega') \cap \mathsf{dom}(\Phi) = \emptyset$, on peut déduire de la bonne formation de $\Omega;\Phi$ que l est aussi dans le domaine de Ω. Le fait que Φ soit une extension loyale de Ω impose alors, d'après la proposition 4.18, que l soit accessible dans Ω. Suivant le même raisonnement que ci-dessus, on obtient donc $l \in \mathcal{A}_{\Omega'}$. Le résultat s'obtient de la même façon pour une hypothèse de la forme $c : \mathsf{C}_{@}l$.

Enfin, $\Omega' <: \Omega$ entraîne alors évidemment $\Omega';\Phi <: \Omega;\Phi$, ce qui achève la preuve du corollaire. □

Le lemme 4.35 permet aussi de conclure que deux systèmes bisimilaires dans un environnement Ω_1 le sont aussi dans un environnement moins « savant », à savoir un sur-type.

Corollaire 4.37 (Renforcement de l'environnement dans la bisimilarité)**.** *Si* $\Omega_1 <: \Omega_2$ *et* $\Omega_1 \vDash M\ {}_{\tilde{a}_M}{\approx^{al}}_{\tilde{a}_N}\ N$ *alors* $\Omega_2 \vDash M\ {}_{\tilde{a}_M}{\approx^{al}}_{\tilde{a}_N}\ N$.

Démonstration. Ce corollaire découle naturellement du lemme 4.35. En effet, si $\mathcal{R}$ est la relation annotée

$$\Omega_2 \vDash M\ {}_{\tilde{a}_M}\mathcal{R}_{\tilde{a}_N}\ N \quad \text{si} \quad \exists \Omega_1 \quad \Omega_1 <: \Omega_2 \quad \Omega_1 \vDash M\ {}_{\tilde{a}_M}{\approx^{al}}_{\tilde{a}_N}\ N$$

pour une action, le raisonnement est le suivant :

$$\Omega_2 \rhd_{\tilde{a}_M} M \xrightarrow{\mu} \Omega_2 \text{ après } \mu \rhd_{\tilde{a}'_M} M' \tag{4.1}$$
$$\Omega_2 \rhd M \xrightarrow{\mathring{\mu}} \Omega_2 \text{ après } \mu \rhd M' \tag{4.2}$$
$$\Omega_1 \rhd M \xrightarrow{\mathring{\mu}} \Omega_1 \text{ après } \mu \rhd M' \tag{4.3}$$
$$\Omega_1 \rhd_{\tilde{a}_M} M \xrightarrow{\mu} \Omega_1 \text{ après } \mu \rhd_{\tilde{a}'_M} M' \tag{4.4}$$
$$\Omega_1 \rhd_{\tilde{a}_N} N \xRightarrow{\hat{\mu}} \Omega_1 \text{ après } \mu \rhd_{\tilde{a}'_N} N' \tag{4.5}$$
$$\Omega_1 \rhd N \xRightarrow{\hat{\mathring{\mu}}'} \Omega_1 \text{ après } \mu \rhd N' \tag{4.6}$$
$$\Omega_2 \rhd N \xRightarrow{\hat{\mathring{\mu}}'} \Omega_2 \text{ après } \mu \rhd N' \tag{4.7}$$
$$\Omega_2 \rhd_{\tilde{a}_N} N \xRightarrow{\hat{\mu}} \Omega_2 \text{ après } \mu \rhd_{\tilde{a}'_N} N' \tag{4.8}$$

où les résultats 4.3 et 4.7 sont obtenus, respectivement, par affaiblissement et renforcement des transitions et où $\mathring{\mu}$ et $\mathring{\mu}'$ sont deux étiquettes de transitions correspondant à l'action μ. Les ensembles $\tilde{a}'_M$ apparaissant dans les lignes 4.4 et 4.1 sont égaux car ils sont tous deux définis par $\tilde{a}_M$ et l'étiquette $\mathring{\mu}$. La même remarque s'applique pour $\tilde{a}'_N$. □

Enfin, intéressons-nous aux commutations que l'on peut opérer dans l'ordre des hypothèses d'un environnement, sans en modifier la signification.

Proposition 4.38 (Commutation d'hypothèses dans un environnement)**.** *Soient deux environnements* Γ_1 *et* Γ_2 *de formes respectives* $\Gamma^{(1)}, s_1 : \mathsf{E}_1, s_2 : \mathsf{E}_2, \Gamma^{(2)}$ *et* $\Gamma^{(1)}, s_2 : \mathsf{E}_2, s_1 : \mathsf{E}_1, \Gamma^{(2)}$ *où* $s_1 \notin \mathsf{fn}(\mathsf{E}_2)$*. Alors* $\Gamma_1 \vdash \mathbf{env}$ *implique* $\Gamma_2 \vdash \mathbf{env}$ *et* $\Gamma_1 \equiv \Gamma_2$*.*

Démonstration. La proposition 2.24 transposée dans le cadre actuel assure $\Gamma^{(1)} \vdash \mathbf{env}$. On prouve alors successivement que $\Gamma^{(1)}, s_2 : \mathsf{E}_2 \vdash \mathbf{env}$ et $\Gamma^{(1)}, s_2 : \mathsf{E}_2, s_1 : \mathsf{E}_1 \vdash \mathbf{env}$. La première preuve est immédiate en considérant les formes que peut prendre E_2 : par exemple si $\mathsf{E}_2 = \mathsf{C}_{@s}$, on sait que $s \neq s_1$ donc $\Gamma^{(1)} \vdash s : \text{LOC}$ découle immédiatement de $\Gamma^{(1)}, s_1 : \mathsf{E}_1 \vdash s : \text{LOC}$. Les conditions de compatibilité sont immédiatement héritées. Le raisonnement est similaire pour $\Gamma^{(1)}, s_2 : \mathsf{E}_2, s_1 : \mathsf{E}_1 \vdash \mathbf{env}$: Les conditions de compatibilité sont hérités de $\Gamma^{(1)}, s_1 : \mathsf{E}_1 \vdash \mathbf{env}$ si $s_1 \neq s_2$ et de $\Gamma^{(1)}, s_2 : \mathsf{E}_2, s_1 : \mathsf{E}_1 \vdash \mathbf{env}$ sinon.

L'équivalence des environnements obtenus est immédiate. □

Comme promis préalablement, ces commutations permettent de définir des notions de formes normales pour les environnements.

Proposition 4.39 (Formes normales d'un environnement)**.** *Pour tout environnement* Γ *bien formé il existe un environnement équivalent* Γ' *de la forme*

$$\begin{array}{l} s_1 : \text{LOC}, \ldots, s_m : \text{LOC}, \\ \qquad u_1 : \tilde{t}_{i_1} \mapsto t_{i_1}, \ldots, u_n : \tilde{t}_{i_n} \mapsto t_{i_n}, \\ \qquad\qquad v_1 : \mathsf{C}_{1@s_{j_1}}, \ldots, v_o : \mathsf{C}_{o@s_{j_o}} \end{array}$$

de sorte que :

- *les* s_k *soient deux à deux distincts ;*
- $u_k = u_{k'}$ *uniquement si* $k = k'$ *ou si* $t_{i_k} \neq t_{i_{k'}}$ *;*
- $v_k = v_{k'}$ *uniquement si* $k = k'$ *ou si* $s_{j_k} \neq s_{j_{k'}}$*.*

Démonstration. On remarque immédiatement que deux environnements composés des mêmes hypothèses dans un ordre différent et tous deux bien formés sont équivalents.

On peut alors appliquer la proposition 4.38 autant de fois qu'il est nécessaire pour transformer tout environnement en un équivalent de la forme

$$s_1 : \text{LOC}, \ldots, s_m : \text{LOC}, u_1 : \mathsf{E}_1, \ldots, u_n : \mathsf{E}_n$$

où $\mathsf{E}_i \neq \text{LOC}$. Puisque la condition pour faire commuter les deux hypothèses $s_1 : \mathsf{E}_1$ et $s_2 : \mathsf{E}_2$ est le fait que $s_1 \notin \mathsf{fn}(\mathsf{E}_2)$, et que les seuls identifiants pouvant apparaître dans les types sont des localités, l'ordre parmi les hypothèses $u_i : \mathsf{E}_i$ est arbitraire. Il en va de même parmi les hypothèses $s_i : \text{LOC}$.

La dernière remarque à faire pour achever cette preuve est le fait que la condition de compatibilité dans la bonne formation de cet environnement implique que tous les types associés à même identifiant sont compatibles. D'après le théorème 2.17 de complétude sous condition, cela signifie que ces types ont une borne inférieure. On constate par ailleurs facilement que les environnements $\Gamma^{(1)}, u : \mathsf{E}_1, u : \mathsf{E}_2, \Gamma^{(2)}$ et $\Gamma^{(1)}, u : (\mathsf{E}_1 \sqcap \mathsf{E}_2), \Gamma^{(2)}$ sont équivalents. □

Cette proposition implique notamment que tout environnement ne contenant que des noms, tel que les environnements dénotant la connaissance d'un observateur dans les équivalences typées, peuvent se récrire de sorte à contenir

exactement une hypothèse sur chacun des noms de leur domaine. On peut même choisir une forme normale particulière parmi toutes celles possibles, l'unicité d'une telle forme simplifiant notablement sa manipulation.

Définition 4.40 (Forme normale naturelle d'un environnement)**.** *La forme normale d'un environnement* Γ *dans laquelle les localités, ainsi que les passeports et les canaux, apparaissent dans le même ordre que leur première occurrence dans* Γ *est appelée* forme normale naturelle *de* Γ*.*

4.7.5 Contextualité de la bisimilarité

Attaquons maintenant les trois propriétés fondamentales pour prouver que la bisimilarité est incluse dans la congruence loyale : vérifions qu'elle est fermée par les trois critères de contextualité. La propriété essentielle parmi ces trois est bien entendue la fermeture par contexte parallèle.

Lemme 4.41 (Fermeture de la bisimulation par extension loyale)**.** *Si* $\Omega \vDash M \ {}_{\tilde{a}_M}\!\approx^{al}_{\tilde{a}_N} N$*, et que* Ω' *est une extension loyale de* Ω *telle que* $\mathsf{dom}(\Omega')$ *ne contienne que des noms inédits alors* $\Omega;\Omega' \vDash M \ {}_{\tilde{a}_M}\!\approx^{al}_{\tilde{a}_N} N$*.*

Démonstration. Le raisonnement pour cette preuve est exactement identique à celui du corollaire 4.37. □

Théorème 4.42 (Fermeture de la bisimulation par contexte parallèle)**.** *Si* $\Omega \vDash M \ {}_{\tilde{a}_M}\!\approx^{al}_{\tilde{a}_N} N$*,* $l \in \mathcal{A}_\Omega$*,* $\Omega \vdash l[\![O]\!]$ *et* $\mathsf{fn}(O) \cap (\tilde{a}_M \cup \tilde{a}_N) = \emptyset$ *alors* $\Omega \vDash M \,|\, l[\![O]\!] \ {}_{\tilde{a}_M}\!\approx^{al}_{\tilde{a}_N} N \,|\, l[\![O]\!]$*.*

Démonstration. Pour prouver que les deux configurations annotées $\Omega \rhd_{\tilde{a}_M} M \,|\, l[\![O]\!]$ et $\Omega \rhd_{\tilde{a}_N} N \,|\, l[\![O]\!]$ sont loyalement bisimilaires, formons une grande relation typée annotée qui les contienne. Cette preuve est ici nettement plus complexe que celles de la littérature pour essentiellement deux raisons : les configurations sont annotées donc les évolutions des annotations doivent être prises en compte ; par ailleurs, la présence de contrôle des observations par passeport impose la conservation de suffisamment de passeports pour exploiter la bisimilarité initialement connue sur les systèmes M et N.

Soit la relation $\mathcal{R}$ telle que

$$\Omega \vDash (\mathsf{new}\,\Phi_M)(M \,|\, \prod_i l_i[\![O_i]\!]) \ {}_{\tilde{a}_M}\mathcal{R}_{\tilde{a}_N} \ (\mathsf{new}\,\Phi_N)(N \,|\, \prod_i l_i[\![O_i]\!])$$

quand il existe Ω_O un environnement et $\mathcal{N}_O$ un ensemble de noms tel que

- $\Omega_O \equiv \Omega_{O(M)};\Omega_{\Phi_M}$ avec $\Omega_{O(M)} <: \Omega$, $\mathsf{dom}(\Omega_{O(M)}) \cap \mathsf{dom}(\Phi_M) = \emptyset$ et $\Phi_M <: \Omega_{\Phi_M}$;
- $\Omega_O \equiv \Omega_{O(N)};\Omega_{\Phi_N}$ avec $\Omega_{O(N)} <: \Omega$, $\mathsf{dom}(\Omega_{O(N)}) \cap \mathsf{dom}(\Phi_N) = \emptyset$ et $\Phi_N <: \Omega_{\Phi_N}$;
- $\Omega_O; p_1 : \star \mapsto l_1, \ldots \vDash M \ {}_{\tilde{a}^i_M}\!\approx^{al}_{\tilde{a}^i_N} N$ où les p_i sont inédits, $\tilde{a}^i_M = (\mathsf{dom}(\Phi_M) \cup \tilde{a}_M) \setminus \mathcal{N}_O$ et $\tilde{a}^i_N = (\mathsf{dom}(\Phi_N) \cup \tilde{a}_N) \setminus \mathcal{N}_O$;
- $(\mathsf{dom}(\Phi_M) \cup \tilde{a}_M) \cap \mathcal{N}_O = (\mathsf{dom}(\Phi_N) \cup \tilde{a}_N) \cap \mathcal{N}_O$;
- $\mathsf{fn}(O_i) \subseteq \mathcal{N}_O$ pour tout i et $\mathsf{dom}(\Omega_O) \subseteq \mathcal{N}_O$;
- $\Omega_O \vdash \prod_i l_i[\![O_i]\!]$.

On notera dans la suite Ω' pour l'environnement $\Omega_O; p_1 : \star \mapsto l_1, \ldots$ L'idée sous-jacente aux environnements Ω_O et Ω' qui apparaissent dans cette définition est de modéliser la connaissance de l'observateur qui a rajouté le contexte $l[\![O]\!]$ et qui se retrouve éclatée à différents endroits dans le système au fur et à mesure des transitions. En particulier, le fragment $\Omega_{O(M)}$ de Ω_O peut contenir des hypothèses concernant les noms $\tilde{a}_M$ encore inconnus pour Ω mais qui peuvent être intervenus dans des communications entre M et $l[\![O]\!]$ et donc être partiellement connus de l'observateur au sens large. De la même façon, l'ensemble de noms $\mathcal{N}_O$ contient l'ensemble des noms qui sont connus de l'observateur. Cela signifie que les noms $(\mathsf{dom}(\Phi_M) \cup \tilde{a}_M) \setminus \mathcal{N}_O$ sont exactement les noms « privés » du système M, ce qui explique que les configurations correspondant aux systèmes M et N doivent être bisimilaires avec les noms de $\mathcal{N}_O$ libres. De plus, l'observateur est capable d'observer certains comportements dans toutes les localités où il dispose d'un processus, c'est-à-dire les localités l_i, même si ces localités ne sont pas explicitement accessible à Ω_O : le processus initialement placé par l'observateur pourrait par exemple créer une nouvelle localité sans disposer de passeport pour y accéder. Il est par conséquent nécessaire de préserver l'hypothèse de bisimilarité aussi au sein de ces localités, d'où l'ajout explicite de passeports à Ω_O pour comparer M et N.

La première remarque à faire sur cette relation typée est bien entendu le fait que $\Omega \vDash M \,|\, l[\![O]\!] \;{}_{\tilde{a}_M}\mathcal{R}_{\tilde{a}_N}\; N \,|\, l[\![O]\!]$. En prenant $\Omega_O = \Omega$ et $\mathcal{N}_O = \mathsf{fn}(O) \cup \mathsf{dom}(\Omega_O)$, le résultat découle simplement du lemme 4.41 puisque $p_l : \star \mapsto l$ est une extension loyale de Ω sachant que $l \in \mathcal{A}_\Omega$.

Prouvons donc que $\mathcal{R}$ est une bisimulation loyale modulo congruence structurelle, ce qui suffira à montrer qu'elle est incluse dans la bisimilarité. Supposons pour cela que la configuration $\Omega \rhd_{\tilde{a}_M} (\mathsf{new}\,\Phi_M)(M \,|\, \prod_i l_i[\![O_i]\!])$ effectue une action μ donc aussi une transition du SLTE. Puisque chaque règle du SLTE ne correspond qu'à certaine forme de système, la structure du système effectuant la transition impose les choix des dernières règles de sa preuve : dans ce cas particulier, un ensemble de (TE-OUV) et (TE-C-NEW) correspondent à $(\mathsf{new}\,\Phi_M)$ puis soit (TE-COMM) soit (TE-C-PAR) pour la composition parallèle entre M et $\prod_i l_i[\![O_i]\!]$. On distingue les trois cas possibles pour cette règle : le cas (TE-COMM) et les deux cas (TE-C-PAR), celui où M effectue l'action, et celui où seul $\prod_i l_i[\![O_i]\!]$ évolue.

(te-c-par), seul M évolue. L'idée dans ce cas est d'utiliser la bisimilarité connue entre M et N pour montrer que N peut aussi effectuer la même action et arriver dans un état équivalent.

Si la transition est invisible, donc de la forme $\xrightarrow{\tau}$, on voit aisément que cette transition est prouvable indépendamment de la connaissance de l'observateur, pour peu que la configuration soit bien formée. Par conséquent, $\Omega' \rhd_{\tilde{a}^i_M} M \xrightarrow{\tau} \Omega' \rhd_{\tilde{a}^i_M} M'$, ce qui entraîne $\Omega' \rhd_{\tilde{a}^i_N} N \stackrel{\hat{\tau}}{\Longrightarrow} \Omega' \rhd_{\tilde{a}^i_N} N'$ avec $\Omega' \vDash M' \;{}_{\tilde{a}^i_M}{\approx^{al}}_{\tilde{a}^i_N}\; N'$. On obtient donc directement

$$\Omega \vDash (\mathsf{new}\,\Phi_M)(M' \,|\, \prod_i l_i[\![O_i]\!]) \;{}_{\tilde{a}_M}\mathcal{R}_{\tilde{a}_N}\; (\mathsf{new}\,\Phi_N)(N' \,|\, \prod_i l_i[\![O_i]\!])$$

Si l'action est une émission $\xrightarrow{(\tilde{b})a!V}$, c'est-à-dire

$$\begin{array}{l}\Omega \rhd_{\tilde{a}_M} (\mathsf{new}\,\Phi_M)(M \,|\, \prod_i l_i[\![O_i]\!]) \\ \quad \xrightarrow{(\tilde{b})a!V} \Omega, \langle V : \Omega^r(a)\rangle \circledast l \rhd_{\tilde{a}_{M'}} (\mathsf{new}\,\Phi_{M'})(M' \,|\, \prod_i l_i[\![O_i]\!])\end{array}$$

cette action suppose qu'il existe une transition correspondante prouvable dans le SLTE. Il existe donc un environnement Φ'_M tel que

$$\begin{array}{l}\Omega \rhd (\mathsf{new}\,\Phi_M)(M \mid \prod_i l_i[\![O_i]\!]) \\ \quad \xrightarrow{(\Phi'_M)a!V} \Omega,\langle V : \Omega^r(a)\rangle \circledcirc l \rhd (\mathsf{new}\,\Phi_{M'})(M' \mid \prod_i l_i[\![O_i]\!])\end{array}$$

avec $\tilde{b} = (\mathsf{dom}(\Phi'_M) \cup \tilde{a}_M) \cap \mathsf{fn}(V)$ et $\tilde{a}_{M'} = (\mathsf{dom}(\Phi'_M) \cup \tilde{a}_M) \setminus \mathsf{fn}(V)$. La preuve de cette transition contient en particulier la preuve de la transition :

$$\Omega \rhd M \xrightarrow{(\Phi''_M)a!V} \Omega,\langle V : \Omega^r(a)\rangle \circledcirc l \rhd M'$$

D'après le corrollaire 4.36, $\Omega' <: \Omega$ et le fait que la configuration $\Omega' \rhd M$ soit bien formée implique que la transition

$$\Omega' \rhd M \xrightarrow{(\Phi''_M)a!V} \Omega',\langle V : \Omega'^r(a)\rangle \circledcirc l \rhd M'$$

est prouvable dans le SLTE. On peut alors en déduire une action de la configuration annotée $\Omega' \rhd_{\tilde{a}^i_M} M$:

$$\Omega' \rhd_{\tilde{a}^i_M} M \xrightarrow{(\tilde{b}')a!V} \Omega',\langle V : \Omega'^r(a)\rangle \circledcirc l \rhd_{\tilde{a}^i_{M'}} M'$$

où $\tilde{b}' = (\mathsf{dom}(\Phi''_M) \cup \tilde{a}^i_M) \cap \mathsf{fn}(V)$ et $\tilde{a}^i_{M'} = (\mathsf{dom}(\Phi''_M) \cup \tilde{a}^i_M) \setminus \mathsf{fn}(V)$. L'hypothèse $\Omega' \vDash M \;{}_{\tilde{a}^i_M}{\approx}^{al}_{\tilde{a}^i_N}\; N$ permet alors de conclure que N peut effectuer la même action, c'est-à-dire :

$$\Omega' \rhd_{\tilde{a}^i_N} N \xRightarrow{(\tilde{b}')a!V} \Omega',\langle V : \Omega'^r(a)\rangle \circledcirc l \rhd_{\tilde{a}^i_{N'}} N'$$

avec

$$\Omega',\langle V : \Omega'^r(a)\rangle \circledcirc l \vDash M' \;{}_{\tilde{a}^i_{M'}}{\approx}^{al}_{\tilde{a}^i_{N'}}\; N'$$

On peut déduire de cette action qu'il existe une transition prouvable dans le SLTE de la forme :

$$\Omega' \rhd N \xRightarrow{(\Phi''_N)a!V} \Omega',\langle V : \Omega'^r(a)\rangle \circledcirc l \rhd N'$$

où $\tilde{b}' = (\mathsf{dom}(\Phi''_N) \cup \tilde{a}^i_N) \cap \mathsf{fn}(V)$ et $\tilde{a}^i_{N'} = (\mathsf{dom}(\Phi''_N) \cup \tilde{a}^i_N) \setminus \mathsf{fn}(V)$. Puisque l'action $(\tilde{b}')a!V$ est activée par l'environnement Ω, le lemme 4.35 permet de conclure que

$$\Omega \rhd N \xRightarrow{(\Phi''_N)a!V} \Omega,\langle V : \Omega^r(a)\rangle \circledcirc l \rhd N'$$

est aussi prouvable dans le SLTE. En appliquant les règles (TE-C-PAR), (TE-C-NEW) et (TE-OUV), on peut étendre cette preuve en

$$\begin{array}{l}\Omega \rhd (\mathsf{new}\,\Phi_N)(N \mid \prod_i l_i[\![O_i]\!]) \\ \quad \xRightarrow{(\Phi'_N)a!V} \Omega,\langle V : \Omega^r(a)\rangle \circledcirc l \rhd (\mathsf{new}\,\Phi_{N'})(N' \mid \prod_i l_i[\![O_i]\!])\end{array}$$

d'où l'on peut déduire l'action

$$\begin{array}{l}\Omega \rhd_{\tilde{a}_N} (\mathsf{new}\,\Phi_N)(N \mid \prod_i l_i[\![O_i]\!]) \\ \quad \xRightarrow{(\tilde{b}'')a!V} \Omega,\langle V : \Omega^r(a)\rangle \circledcirc l \rhd_{\tilde{a}_{N'}} (\mathsf{new}\,\Phi_{N'})(N' \mid \prod_i l_i[\![O_i]\!])\end{array}$$

où $\tilde{b}'' = (\mathsf{dom}(\Phi'_N) \cup \tilde{a}_N) \cap \mathsf{fn}(V)$ et $\tilde{a}_{N'} = (\mathsf{dom}(\Phi'_N) \cup \tilde{a}_N) \setminus \mathsf{fn}(V)$.
Il s'agit maintenant de vérifier que $\tilde{b}'' = \tilde{b}$, donc que les deux actions sont effectivement identiques, et que toutes les hypothèses sont conservées.

$$\begin{array}{rcl}
\tilde{b}'' & = & (\mathsf{dom}(\Phi'_N) \cup \tilde{a}_N) \cap \mathsf{fn}(V) \\
& = & (\mathsf{dom}(\Phi'_N) \cap \mathsf{fn}(V)) \cup (\tilde{a}_N \cap \mathsf{fn}(V)) \\
& = & ((\mathsf{dom}(\Phi_N) \cap \mathcal{N}_O \cap \mathsf{fn}(V)) \cup \\
& & \quad ((\mathsf{dom}(\Phi_N) \setminus \mathcal{N}_O) \cap \mathsf{fn}(V)) \cup (\mathsf{dom}(\Phi''_N) \cap \mathsf{fn}(V))) \cup \\
& & \quad ((\tilde{a}_N \cap \mathcal{N}_O \cap \mathsf{fn}(V)) \cup ((\tilde{a}_N \setminus \mathcal{N}_O) \cap \mathsf{fn}(V))) \\
& = & (\mathsf{dom}(\Phi_N) \cap \mathcal{N}_O \cap \mathsf{fn}(V)) \cup (\tilde{a}_N \cap \mathcal{N}_O \cap \mathsf{fn}(V)) \cup \\
& & \quad (\mathsf{dom}(\Phi''_N) \cap \mathsf{fn}(V))) \cup (\tilde{a}^i_N \cap \mathsf{fn}(V)) \\
& = & ((\mathsf{dom}(\Phi_N) \cup \tilde{a}_N) \cap \mathcal{N}_O \cap \mathsf{fn}(V)) \cup \tilde{b}' \\
& = & ((\mathsf{dom}(\Phi_M) \cup \tilde{a}_M) \cap \mathcal{N}_O \cap \mathsf{fn}(V)) \cup \tilde{b}' \\
& = & \tilde{b}
\end{array}$$

Quant à la vérification des hypothèses, il faut donner un environnement Ω'_O. Assez naturellement, on prend $\Omega_O \ \mathsf{après}\ \mu = \Omega_O, \langle V : \Omega^r_O(a) \rangle \circledcirc l$.

$$\Omega', \big\langle V : {\Omega'}^r(a) \big\rangle \circledcirc l \vDash M' \ {}_{\tilde{a}^i_{M'}} \approx^{al}_{\tilde{a}^i_{N'}} \ N'$$

devient alors

$$\Omega_O, \langle V : {\Omega_O}^r(a) \rangle \circledcirc l; p_1 : \star \mapsto l_1, \ldots \vDash M' \ {}_{\tilde{a}^i_{M'}} \approx^{al}_{\tilde{a}^i_{N'}} \ N'$$

puisque le fragment $p_1 : \ldots$ de Ω' ne contient aucune hypothèse concernant le canal a ce qui autorise l'utilisation du lemme 4.38.
Naturellement, on prend aussi $\mathcal{N}'_O = \mathcal{N}_O \cup \mathsf{fn}(V)$. Prouvons alors que $\tilde{a}^i_{M'}$, tel que défini par l'action effectuée par la configuration annotée $\Omega' \rhd_{\tilde{a}^i_M} M$, correspond effectivement à $(\mathsf{dom}(\Phi_{M'}) \cup \tilde{a}_{M'}) \setminus \mathcal{N}'_O$. En effet

$$\begin{array}{rcl}
\tilde{a}^i_{M'} & = & (\mathsf{dom}(\Phi''_M) \cup \tilde{a}^i_M) \setminus \mathsf{fn}(V) \\
& = & (\mathsf{dom}(\Phi''_M) \cup ((\mathsf{dom}(\Phi_M) \cup \tilde{a}_M) \setminus \mathcal{N}_O)) \setminus \mathsf{fn}(V) \\
& = & (\mathsf{dom}(\Phi''_M) \setminus \mathsf{fn}(V)) \cup (\mathsf{dom}(\Phi_M) \setminus \mathcal{N}'_O) \cup (\tilde{a}_M \setminus \mathcal{N}'_O) \\
& = & (\mathsf{dom}(\Phi''_M) \setminus \mathcal{N}'_O) \cup (\mathsf{dom}(\Phi_M) \setminus \mathsf{dom}(\Phi_{M'})) \setminus \mathcal{N}'_O) \\
& & \quad \cup (\mathsf{dom}(\Phi_{M'}) \setminus \mathcal{N}'_O) \cup (\tilde{a}_M \setminus \mathcal{N}'_O) \\
& = & (\mathsf{dom}(\Phi'_M) \setminus \mathcal{N}'_O) \cup (\tilde{a}_M \setminus \mathcal{N}'_O) \cup (\mathsf{dom}(\Phi_{M'}) \setminus \mathcal{N}'_O) \\
& = & (\tilde{a}_{M'} \setminus \mathcal{N}'_O) \cup (\mathsf{dom}(\Phi_{M'}) \setminus \mathcal{N}'_O)
\end{array}$$

Ce raisonnement peut aussi être tenu pour les noms correspondant associés au système N.
On vérifie aussi que l'invariant entre les deux systèmes comparés, l'égalité entre $(\mathsf{dom}(\Phi_M) \cup \tilde{a}_M) \cap \mathcal{N}_O$ et $(\mathsf{dom}(\Phi_N) \cup \tilde{a}_N) \cap \mathcal{N}_O$ est conservé.

$$\begin{array}{l}
(\mathsf{dom}(\Phi_{M'}) \cup \tilde{a}_{M'}) \cap \mathcal{N}'_O \\
\quad = ((\mathsf{dom}(\Phi_M) \setminus \mathsf{dom}(\Phi'_M)) \cup ((\tilde{a}_M \cup \mathsf{dom}(\Phi'_M)) \setminus \mathsf{fn}(V))) \cap \mathcal{N}'_O \\
\quad = ((\mathsf{dom}(\Phi_M) \setminus \mathsf{fn}(V) \setminus \mathsf{dom}(\Phi'_M)) \cup (\mathsf{dom}(\Phi'_M) \setminus \mathsf{fn}(V)) \\
\qquad \cup ((\mathsf{dom}(\Phi_M) \cap \mathsf{fn}(V)) \setminus \mathsf{dom}(\Phi'_M)) \cup (\tilde{a}_M \setminus \mathsf{fn}(V))) \cap \mathcal{N}'_O \\
\quad = ((\mathsf{dom}(\Phi_M) \setminus \mathsf{fn}(V)) \cup (\tilde{a}_M \setminus \mathsf{fn}(V)) \cup (\mathsf{dom}(\Phi'_M) \setminus \mathsf{fn}(V)) \\
\qquad \cup ((\mathsf{dom}(\Phi_M) \cap \mathsf{fn}(V)) \setminus \mathsf{dom}(\Phi'_M))) \cap \mathcal{N}'_O \\
\quad = ((\mathsf{dom}(\Phi_M) \setminus \mathsf{fn}(V)) \cup (\tilde{a}_M \setminus \mathsf{fn}(V))) \cap \mathcal{N}'_O \\
\quad = ((\mathsf{dom}(\Phi_M) \setminus \mathsf{fn}(V)) \cup (\tilde{a}_M \setminus \mathsf{fn}(V))) \cap \mathcal{N}_O
\end{array}$$

car :

– $\mathcal{N}'_O = \mathcal{N}_O \cup \mathsf{fn}(V)$ entraîne que $(\mathsf{dom}(\Phi'_M) \setminus \mathsf{fn}(V)) \cap \mathcal{N}'_O$ est inclus dans $(\mathsf{dom}(\Phi'_M) \setminus \mathsf{fn}(V)) \cap \mathcal{N}_O$, lui-même inclus dans $(\mathsf{dom}(\Phi_M) \setminus \mathsf{fn}(V)) \cap \mathcal{N}_O$;
– $(\mathsf{dom}(\Phi_M) \cap \mathsf{fn}(V)) \setminus \mathsf{dom}(\Phi'_M) = \emptyset$ car tous les noms de Φ_M apparaissant dans V doivent être ouverts et sont donc mentionnés dans Φ'_M.

Le même raisonnement pour N aboutit à :

$$\begin{array}{rcl}(\mathsf{dom}(\Phi_{M'}) \cup \tilde{a}_{M'}) \cap \mathcal{N}'_O & = & ((\mathsf{dom}(\Phi_M) \cup \tilde{a}_M) \cap \mathcal{N}_O) \setminus \mathsf{fn}(V) \\ & = & ((\mathsf{dom}(\Phi_N) \cup \tilde{a}_N) \cap \mathcal{N}_O) \setminus \mathsf{fn}(V) \\ & = & (\mathsf{dom}(\Phi_{N'}) \cup \tilde{a}_{N'}) \cap \mathcal{N}'_O\end{array}$$

Considérons maintenant si Ω'_O peut être décomposé en $\Omega'_{O(M)}; \Omega_{\Phi_{M'}}$. Puisque Φ_M est un environnement de la forme $a_1 : \mathsf{E}_1; \ldots$ c'est-à-dire constitué par la concaténation d'hypothèses $a_i : \mathsf{E}_i$ où les noms a_i sont distincts deux à deux et que $\Phi_M <: \Omega_{\Phi_M}$, la forme normale naturelle de Ω_{Φ_M} est forcément de la forme $a_{i_1} : \mathsf{E}'_{i_1}, \ldots$ où pour tout j, $\mathsf{E}_{i_j} <: \mathsf{E}'_{i_j}$. Par ailleurs Φ_M est décomposé par la transition en l'environnement équivalent $\Phi^o_M; \Phi_{M'}$ où Φ^o_M est la partie de Φ_M qui est ouverte par la transition, puisque la preuve de la transition implique que les noms apparaissant dans $\Phi_{M'}$ ne sont pas mentionnés dans les types de ceux ouverts. Puisque les noms mentionnés dans un sur-type sont tous mentionnés dans un sous-type (les types C sont totalement clos) et que $\mathsf{E}_{i_j} <: \mathsf{E}'_{i_j}$, on peut conclure que Ω_{Φ_M} est aussi équivalent à $\Omega_{\Phi^o_M}; \Omega_{\Phi_{M'}}$ où, comme la notation l'indique, $\Phi^o_M <: \Omega_{\Phi^o_M}$ et $\Phi_{M'} <: \Omega_{\Phi_{M'}}$, ces deux parties étant obtenues en séparant les hypothèses de $a_{i_1} : \mathsf{E}'_{i_1}, \ldots$ de la même façon que Φ_M est décomposé en $\Phi^o_M; \Phi_{M'}$. Alors $\Omega_O \equiv \Omega_{O(M)}; \Omega_{\Phi^o_M}; \Omega_{\Phi_{M'}}$, et

$$\Omega_O, \langle V : \Omega^r_O(a) \rangle \circledcirc l \equiv ((\Omega_{O(M)}; \Omega_{\Phi^o_M}), \langle V : (\Omega_{O(M)}; \Omega_{\Phi^o_M})^r(a) \rangle \circledcirc l); \Omega_{\Phi_{M'}}$$

car les noms apparaissant dans V ne peuvent être dans le domaine de $\Omega_{\Phi_{M'}}$ sans quoi leur portée devrait être ouverte par la transition ; qui plus est, $\Omega_{\Phi_{M'}}$ ne peut pas contenir d'hypothèse concernant le nom a car toutes ses hypothèses portent sur des noms liés dans le système. Cette fois encore, un raisonnement identique peut être tenu pour le système N.

Enfin, $\Omega_O \vdash \prod_i l_i[\![O_i]\!]$ implique $\Omega_O, \langle V : \Omega^r_O(a) \rangle \circledcirc l \vdash \prod_i l_i[\![O_i]\!]$ d'après la proposition 4.3. La preuve que

$$\begin{array}{l}\Omega \text{ après } (\tilde{b})a!V \vDash \\ \quad (\mathsf{new}\, \Phi_{M'})(M' \mid \prod_i l_i[\![O_i]\!]) \; \tilde{a}_{M'} \mathcal{R} \tilde{a}_{N'} \; (\mathsf{new}\, \Phi_{N'})(N' \mid \prod_i l_i[\![O_i]\!])\end{array}$$

est donc achevée.

Supposons donc pour finir ce cas que M effectue une réception, soit une transition

$$\Omega \rhd (\mathsf{new}\, \Phi_M)(M \mid \prod_i l_i[\![O_i]\!]) \xrightarrow{(\Phi)a?V} \Omega; \Phi \rhd (\mathsf{new}\, \Phi_M)(M' \mid \prod_i l_i[\![O_i]\!])$$

dans le SLTE. Les règles apparaissant sous (TE-C-PAR) qui correspond à la composition parallèle de M et de $\prod_i l_i[\![O_i]\!]$ peuvent être soit des (TE-C-NEW) soit des (TE-AFF). Or une règle (TE-C-NEW) suivie d'une règle (TE-AFF) comme

$$\dfrac{\dfrac{\Omega; \Phi \rhd M \xrightarrow{(\Phi')a?V} \Omega; \Phi; \Phi' \rhd M'}{\Omega; \Phi \rhd (\mathsf{new}\, b : \mathsf{E})\, M \xrightarrow{(\Phi')a?V} \Omega; \Phi; \Phi' \rhd (\mathsf{new}\, b : \mathsf{E})\, M'}}{\Omega \rhd (\mathsf{new}\, b : \mathsf{E})\, M \xrightarrow{(\Phi;\Phi')a?V} \Omega; \Phi; \Phi' \rhd (\mathsf{new}\, b : \mathsf{E})\, M'}$$

peut commuter avec elle pour donner

$$\frac{\dfrac{\Omega;\Phi \rhd M \xrightarrow{(\Phi')a?V} \Omega;\Phi;\Phi' \rhd M'}{\Omega \rhd M \xrightarrow{(\Phi;\Phi')a?V} \Omega;\Phi;\Phi' \rhd M'}}{\Omega \rhd (\mathsf{new}\, b : \mathsf{E})\, M \xrightarrow{(\Phi;\Phi')a?V} \Omega;\Phi;\Phi' \rhd (\mathsf{new}\, b : \mathsf{E})\, M'}$$

car la convention de Barendregt garantit que les noms de Φ n'entrent pas en collision avec b. On obtient donc une preuve de

$$\Omega \rhd M \xrightarrow{(\Phi)a?V} \Omega;\Phi \rhd M'$$

le cas échéant par une suite de telles commutations. Par affaiblissement, on en déduit que

$$\Omega' \rhd M \xrightarrow{(\Phi)a?V} \Omega';\Phi \rhd M'$$

Puisque $\Omega' \vDash M \ {}_{\tilde{a}^i_M}{\approx}^{al}_{\tilde{a}^i_N}\ N$, on obtient une transition

$$\Omega' \rhd N \xRightarrow{(\Phi)a?V} \Omega';\Phi \rhd N'$$

qui peut être renforcée en

$$\Omega \rhd N \xRightarrow{(\Phi)a?V} \Omega;\Phi \rhd N'$$

d'où l'on conclut aisément

$$\Omega \rhd (\mathsf{new}\, \Phi_N)(N \mid \prod_i l_i[\![O_i]\!]) \xRightarrow{(\Phi)a?V} \Omega;\Phi \rhd (\mathsf{new}\, \Phi_N)(N' \mid \prod_i l_i[\![O_i]\!])$$

On sait par ailleurs que $\Omega';\Phi \vDash M' \ {}_{\tilde{a}^i_M}{\approx}^{al}_{\tilde{a}^i_N}\ N'$. En prenant simplement $\Omega_O;\Phi$ comme environnement Ω'_O et $\mathcal{N}_O \cup \mathsf{dom}(\Phi)$ comme ensemble $\mathcal{N}'_O$, on voit facilement que toutes les hypothèses sont conservées, notamment parce que $\Omega_O;\Phi \equiv \Omega_{O(M)};\Phi;\Omega_{\Phi_M}$, etc.

(te-c-par), seul $\prod_i l_i[\![O_i]\!]$ évolue. Dans ce cas, il paraît assez naturel que les actions effectuées par le système contenant M seront aisément simulées par la même action dans celui contenant N, et réciproquement.

Considérons les différentes actions que peut effectuer $\prod_i l_i[\![O_i]\!]$. Si cette action est invisible, la preuve de la transition contient l'une des règles suivantes :

- (te-comm), (te-if-v), (te-if-f), (te-here) et (te-rec) : le théorème 4.13 de préservation du typage garantit que l'hypothèse $\Omega_O \vdash \prod_i l_i[\![O_i]\!]$ est conservée, toutes les autres hypothèses étant intouchées par ces réductions ;
- (te-goto) : la transition effectuée est $\Omega_O \rhd l_{i_0}[\![\mathsf{goto}_p\, k.\ P]\!]$; on peut conclure de $\Omega_O \vdash \prod_i l_i[\![O_i]\!]$ que $\Omega_O \vdash p : l_{i_0} \mapsto k$, donc $k \in \mathcal{A}_{\Omega'}$; le lemme 4.41 permet alors de conclure que, si p_k est un nom inédit, $\Omega';p_k : \star \mapsto k \vDash M \ {}_{\tilde{a}^i_M}{\approx}^{al}_{\tilde{a}^i_N}\ N$; par ailleurs, d'après le corollaire 4.37

$$\Omega_O;p_1 : \star \mapsto l_1, \ldots, p_{i_0-1} : \star \mapsto l_{i_0-1}, p_k : \star \mapsto k, p_{i_0+1} : \star \mapsto l_{i_0+1}, \ldots \vDash M \ {}_{\tilde{a}^i_M}{\approx}^{al}_{\tilde{a}^i_N}\ N$$

- (te-split) et (te-rep) : le raisonnement est très similaire, la modification de Ω' n'étant par ailleurs due qu'à la correspondance un pour un entre les processus de l'observateur et les passeports de l'environnement ;

– (TE-NEWLOC) : alors la conclusion de cette règle est de la forme

$$\begin{array}{c}\Omega \rhd l_{i_0}[\![\mathsf{newloc}\, k, (\tilde{c}), (\tilde{p}), (\tilde{q}) : \sum x : \text{LOC}.\ \mathsf{T}\ \mathsf{with}\ P_k\ \mathsf{in}\ P]\!] \\ \xrightarrow{\tau} \Omega \rhd (\mathsf{new}\langle k, ((\tilde{c}), (\tilde{p}), (\tilde{q})) : \mathsf{T}\{{}^{l}/_{x}\}\rangle)\, k[\![P_k]\!] \mid l_{i_0}[\![P]\!]\end{array}$$

dans laquelle apparaît, en notant $\Phi = \langle k, ((\tilde{c}), (\tilde{p}), (\tilde{q})) : \mathsf{T}\{{}^{l}/_{x}\}\rangle$, un $(\mathsf{new}\,\Phi)$ qui modifie la structure du système de l'observateur ; on utilise donc le fait que $\mathcal{R}$ est une bisimulation modulo congruence structurelle pour traiter ce cas. En effet, par affaiblissement, on obtient

$$\Omega_O; \Phi \vdash \prod_{i \neq i_0} l_i[\![O_i]\!] \mid k[\![P_k]\!] \mid l_{i_0}[\![P]\!]$$

Par ailleurs $\Phi; p : \star \mapsto k$ est une extension loyale de Ω' car tous les nouveaux canaux apportés par cette extension sont localisés dans la nouvelle localité k, et tous les passeports qu'elle contient mènent à l_{i_0}, localité déjà accessible dans Ω', ou à k. Par conséquent le lemme 4.41 impose que

$$\Omega'; \Phi; p : \star \mapsto k \vDash M\ {}_{\tilde{a}^i_M}{\approx}^{al}_{\tilde{a}^i_N}\ N$$

En prenant donc $\Omega'_O = \Omega_O; \Phi$, $\mathcal{N}'_O = \mathcal{N}_O \cup \mathsf{dom}(\Phi)$ et les systèmes $(\mathsf{new}\,\Phi_M; \Phi)(M \mid \prod_i l_i[\![O_i]\!])$ et $(\mathsf{new}\,\Phi_N; \Phi)(N \mid \prod_i l_i[\![O_i]\!])$ sans modifier nullement $\tilde{a}_M$ et $\tilde{a}_N$, toutes les hypothèses sont vérifiées :

- Ω_O peut se décomposer en $\Omega_{O(M)}; (\Omega_{\Phi_M}; \Phi)$ où $\Omega_{\Phi_M}; \Phi$ est clairement un sur-type de $\Phi_M; \Phi$; un découpage similaire s'applique pour N ;
- $(\mathsf{dom}(\Phi_M; \Phi) \cup \tilde{a}_M) \backslash \mathcal{N}'_O = (\mathsf{dom}(\Phi_M) \cup \tilde{a}_M) \backslash \mathcal{N}_O$ donc les ensembles de noms $\tilde{a}^i_M$ et $\tilde{a}^i_N$ ne sont pas modifiés dans cette nouvelle configuration ;
- $(\mathsf{dom}(\Phi_M; \Phi) \cup \tilde{a}_M) \cap \mathcal{N}'_O = ((\mathsf{dom}(\Phi_M) \cup \tilde{a}_M) \cap \mathcal{N}_O) \cup \mathsf{dom}(\Phi)$ donc l'égalité avec $(\mathsf{dom}(\Phi_N; \Phi) \cup \tilde{a}_N) \cap \mathcal{N}'_O$ est conservée ;
- $\mathsf{fn}(O_i) \subseteq \mathcal{N}'_O$ pour tout i et $\mathsf{dom}(\Omega_O; \Phi) \subseteq \mathcal{N}'_O$.

– enfin (TE-NEWCHAN) et (TE-NEWPASS) sont similaires au cas précédent.

Il nous reste à envisager le cas où $\prod_i l_i[\![O_i]\!]$ effectue une action visible. Si

$$\Omega \rhd_{\tilde{a}_M} (\mathsf{new}\,\Phi_M)(M \mid \prod_{i<i_0} l_i[\![O_i]\!] \mid l_{i_0}[\![c\,!\,\langle V\rangle\, O'_{i_0}]\!] \mid \prod_{i>i_0} l_i[\![O_i]\!])$$

effectue une action $(\tilde{a})c!V$ pour devenir

$$\Omega \rhd_{\tilde{a}'_M} (\mathsf{new}\,\Phi'_M)(M \mid \prod_{i<i_0} l_i[\![O_i]\!] \mid l_{i_0}[\![O'_{i_0}]\!] \mid \prod_{i>i_0} l_i[\![O_i]\!])$$

on sait que le système contenant N peut effectuer une action similaire $(\tilde{a}')c!V$. Afin de vérifier qu'il s'agit de la même action, il suffit de constater que $\tilde{a}$ et $\tilde{a}'$ doivent être égaux. Or, si la transition qui prouve cette action est étiquetée par $(\Phi''_M)c!V$, $\tilde{a}$ est $(\tilde{a}_M \cup \mathsf{dom}(\Phi''_M)) \cap \mathsf{fn}(V)$, qui doit coïncider avec $(\tilde{a}_M \cup \mathsf{dom}(\Phi_M)) \cap \mathsf{fn}(V)$ car les noms de $\mathsf{dom}(\Phi_M) \cap \mathsf{fn}(V)$ doivent être ouverts donc être dans $\mathsf{dom}(\Phi''_M)$ et que $\mathsf{dom}(\Phi''_M) \subseteq \mathsf{dom}(\Phi_M)$ parce que les seuls noms liés sont Φ_M. De plus $\mathsf{fn}(V) \subseteq \mathsf{fn}(O_{i_0}) \subseteq \mathcal{N}_O$ donc

$$\begin{array}{rcl}\tilde{a} & = & (\tilde{a}_M \cup \mathsf{dom}(\Phi_M)) \cap \mathcal{N}_O \cap \mathsf{fn}(V) \\ & = & (\tilde{a}_N \cup \mathsf{dom}(\Phi_N)) \cap \mathcal{N}_O \cap \mathsf{fn}(V) \\ & = & \tilde{a}'\end{array}$$

Bien entendu, l'environnement qui assure la conservation des hypothèses reste Ω_O, en modifiant les découpages en $\Omega'_{O(M)}; \Omega'_{\Phi_M}$ et $\Omega'_{O(N)}; \Omega'_{\Phi_N}$; assez naturellement, avec les noms ouverts par cette émission, $\Omega_O <: \Omega, \langle V : \Omega^r(c)\rangle \circledast l_{i_0}$ permet de prouver $\Omega'_{O(M)} <: \Omega, \langle V : \Omega^r(c)\rangle \circledast l_{i_0}$. De façon similaire, on prend $\mathcal{N}'_O = \mathcal{N}_O$. Considérons

$$\begin{array}{lcl}
(\mathsf{dom}(\Phi'_M) \cup \tilde{a}'_M) \cap \mathcal{N}_O & & \\
\quad = & ((\mathsf{dom}(\Phi_M) \setminus \mathsf{dom}(\Phi''_M)) \cup ((\mathsf{dom}(\Phi''_M) \cup \tilde{a}_M) \setminus \mathsf{fn}(V))) \cap \mathcal{N}_O \\
\quad = & ((\mathsf{dom}(\Phi_M) \setminus \mathsf{dom}(\Phi''_M) \setminus \mathsf{fn}(V) \cup \mathsf{dom}(\Phi''_M) \setminus \mathsf{fn}(V)) \cup \\
& \quad \tilde{a}_M \setminus \mathsf{fn}(V) \cup (\mathsf{dom}(\Phi_M) \cap \mathsf{fn}(V) \setminus \mathsf{dom}(\Phi''_M))) \cap \mathcal{N}_O \\
\quad = & (\mathsf{dom}(\Phi_M) \cup \tilde{a}_M) \cap \mathcal{N}_O \setminus \mathsf{fn}(V)
\end{array}$$

car $\mathsf{dom}(\Phi_M) \cap \mathsf{fn}(V) \setminus \mathsf{dom}(\Phi''_M) = \emptyset$ et $\mathsf{dom}(\Phi''_M) \subseteq \mathsf{dom}(\Phi_M)$ c'est-à-dire $\mathsf{dom}(\Phi_M) \setminus \mathsf{dom}(\Phi''_M) \cup \mathsf{dom}(\Phi''_M) = \mathsf{dom}(\Phi_M)$. En tenant le même raisonnement pour N et en constatant que

$$(\mathsf{dom}(\Phi_M) \cup \tilde{a}_M) \cap \mathcal{N}_O = (\mathsf{dom}(\Phi_N) \cup \tilde{a}_N) \cap \mathcal{N}_O$$

on obtient le résultat attendu

$$(\mathsf{dom}(\Phi'_M) \cup \tilde{a}'_M) \cap \mathcal{N}_O = (\mathsf{dom}(\Phi'_N) \cup \tilde{a}'_N) \cap \mathcal{N}_O$$

La dernière vérification à effectuer est le fait que les ensembles $\tilde{a}^i_M$ et $\tilde{a}^i_N$ sont laissés invariants par cette action. Le raisonnement ressemble encore une fois aux précédents, donc on ne donne que quelques étapes rapides pour M :

$$\begin{array}{lcl}
\tilde{a}^i_M{}' & = & (\mathsf{dom}(\Phi'_M) \cup \tilde{a}'_M) \setminus \mathcal{N}_O \\
& = & (\mathsf{dom}(\Phi_M) \setminus \mathsf{dom}(\Phi''_M) \cup \mathsf{dom}(\Phi''_M) \cup \tilde{a}_M) \setminus \mathsf{fn}(V) \setminus \mathcal{N}_O \\
& = & (\mathsf{dom}(\Phi_M) \cup \tilde{a}_M) \setminus \mathcal{N}_O \\
& = & \tilde{a}^i_M
\end{array}$$

Le résultat est donc prouvé si l'observateur émet un message.

Enfin, si l'observateur reçoit un message par une action $(\Phi)c?V$, bien entendu le système contenant N est aussi capable d'effectuer cette action. Vérifions que les hypothèses sont conservées. Φ est une extension loyale de Ω par conséquent aussi une extension loyale de Ω', toujours suivant la convention de Barendregt. On pose $\mathcal{N}'_O = \mathcal{N}_O \cup \mathsf{dom}(\Phi)$ et $\Omega'_O = \Omega_O; \Phi$, puisque Φ est une extension loyale de Ω et qu'elle ne peut contenir aucun nom de Φ_M et Φ_N libre. Donc Ω'_O peut être décomposé en $\Omega_{O(M)}; \Phi; \Omega_{\Phi_M}$, en respectant les hypothèses. Par ailleurs, le fait que les noms de Φ soient tous inédits garantit que les ensembles $\tilde{a}^i_M$, $\tilde{a}^i_N$, $(\mathsf{dom}(\Phi_M) \cup \tilde{a}_M) \cap \mathcal{N}'_O$ et $(\mathsf{dom}(\Phi_N) \cup \tilde{a}_N) \cap \mathcal{N}'_O$ soient inchangés. Enfin $\Omega_O; \Phi \vdash \prod_i l_i[\![O'_i]\!]$ achève de prouver ce cas.

(te-comm), M et $\prod_i l_i[\![O_i]\!]$ communiquent. La transition

$$\Omega \rhd (\mathsf{new}\,\Phi_M)(M \mid \prod_i l_i[\![O_i]\!]) \xrightarrow{\tau} \Omega \rhd (\mathsf{new}\,\Phi_M)\,(\mathsf{new}\,\Phi'_M)(M' \mid \prod_i l_i[\![O'_i]\!])$$

peut être prouvée avec M ou $\prod_i l_i[\![O_i]\!]$ effectuant l'émission.

Supposons tout d'abord que M émette le message. Donc la transition précédente repose sur les deux transitions suivantes : $\Omega_1 \rhd M \xrightarrow{(\Phi'_M)c!V} \Omega'_1 \rhd$

M' et $\Omega_2 \rhd \prod_i l_i[\![O_i]\!] \xrightarrow{(\Phi'_M)c?V} \Omega'_2 \rhd \prod_i l_i[\![O'_i]\!]$. On déduit de la seconde transition que l est un certain l_{i_0}. Par ailleurs, $\Omega_O \vdash \prod_i l_i[\![O_i]\!]$ implique alors que $\Omega_O \vdash_{l_{i_0}} c : \mathrm{R}\langle\mathsf{T}\rangle$. Les deux conditions d'activation de l'action associée à la transition de M permettent de conclure avec le lemme 4.35 que $\Omega' \rhd M \xrightarrow{(\Phi'_M)c!V} \Omega', \langle V : \Omega'^r(c)\rangle \circledast l \rhd M'$ soit l'action $\Omega' \rhd_{\tilde{a}^i_M} M \xrightarrow{(\tilde{b})c!V} \Omega', \langle V : \Omega'^r(c)\rangle \circledast l \rhd_{\tilde{a}^i_{M'}} M'$. La bisimilarité de M et N permet alors de conclure qu'il existe une transition $\Omega' \rhd N \xrightarrow{(\Phi'_N)c!V} \Omega', \langle V : \Omega'^r(c)\rangle \circledast l \rhd N'$, avec $\tilde{b} = (\mathsf{dom}(\Phi'_M) \cup \tilde{a}^i_M) \cap \mathsf{fn}(V) = (\mathsf{dom}(\Phi'_N) \cup \tilde{a}^i_N) \cap \mathsf{fn}(V)$. Par définition de Φ'_M et de $\tilde{a}^i_M$, on sait donc que $\tilde{b} \cap \mathcal{N}_O = \emptyset$. Cela garantit que $\prod_i l_i[\![O_i]\!]$ soit aussi capable de communiquer avec N pour évoluer vers le même $\prod_i l_i[\![O'_i]\!]$.

On pose $\Omega'_O = \Omega_O, \langle V : \Omega^r_O(c)\rangle \circledast l_{i_0}$ et $\mathcal{N}'_O = \mathcal{N}_O \cup \tilde{b}$. Vérifions que toutes les hypothèses sont maintenues. L'environnement $\langle V : \Omega^r_O(c)\rangle \circledast l_{i_0}$ peut être découpé en deux parties : $\langle V : \Omega^r_O(c)\rangle \circledast l_{i_0} \equiv \Omega_{V_1}; \Omega_{V_2}$. Ω_{V_1} ne contient que des noms qui étaient libres dans le système complet ou apparaissant dans $\tilde{a}_M$ et Ω_{V_2} des noms liés donc apparaissant dans Φ_M ou Φ'_M. Cette découpe est possible car les premiers ne peuvent pas faire référence aux seconds. On peut donc découper Ω'_O en $\Omega_{O(M)}, \Omega_{V_1}; \Omega_{\Phi_M}, \Omega_{V_2}$ avec, naturellement, $\Omega_{O(M)}, \Omega_{V_1} <: \Omega$ et $\Phi_M; \Phi'_M <: \Omega_{\Phi_M}, \Omega_{V_2}$. Vérifions maintenant que $\tilde{a}^i_{M'}$ coïncide avec $(\mathsf{dom}(\Phi_M) \cup \mathsf{dom}(\Phi'_M) \cup \tilde{a}'_M) \setminus \mathcal{N}'_O$.

$$\begin{array}{rcl} \tilde{a}^i_{M'} & = & (\mathsf{dom}(\Phi'_M) \cup \tilde{a}^i_M) \setminus \mathsf{fn}(V) \\ & = & (\mathsf{dom}(\Phi'_M) \cup \mathsf{dom}(\Phi_M) \cup \tilde{a}_M) \setminus \mathcal{N}_O \setminus \mathsf{fn}(V) \\ & = & (\mathsf{dom}(\Phi'_M) \cup \mathsf{dom}(\Phi_M) \cup \tilde{a}'_M) \setminus \mathcal{N}'_O \end{array}$$

Enfin, vérifions l'égalité $(\mathsf{dom}(\Phi_M) \cup \mathsf{dom}(\Phi'_M) \cup \tilde{a}'_M) \cap \mathcal{N}'_O = (\mathsf{dom}(\Phi_N) \cup \mathsf{dom}(\Phi'_N) \cup \tilde{a}'_N) \cap \mathcal{N}'_O$.

$$\begin{array}{rcl} & & (\mathsf{dom}(\Phi_M) \cup \mathsf{dom}(\Phi'_M) \cup \tilde{a}'_M) \cap \mathcal{N}'_O \\ & = & (\mathsf{dom}(\Phi_M) \cup \mathsf{dom}(\Phi'_M) \cup \tilde{a}'_M) \cap (\mathcal{N}_O \cup \tilde{b}) \\ & = & ((\mathsf{dom}(\Phi_M) \cup \tilde{a}_M) \cap \mathcal{N}_O) \cup (\mathsf{dom}(\Phi'_M) \cap \mathcal{N}_O) \\ & & \cup((\mathsf{dom}(\Phi_M) \cup \mathsf{dom}(\Phi'_M) \cup \tilde{a}'_M) \cap \tilde{b}) \end{array}$$

De ces trois composantes, la première possède un équivalent pour N, la deuxième est vide et la troisième est naturellement $\tilde{b}$. On peut donc conclure simplement que

$$(\mathsf{dom}(\Phi_M) \cup \mathsf{dom}(\Phi'_M) \cup \tilde{a}'_M) \cap \mathcal{N}'_O = (\mathsf{dom}(\Phi_N) \cup \mathsf{dom}(\Phi'_N) \cup \tilde{a}'_N) \cap \mathcal{N}'_O$$

Par ailleurs, les hypothèses $\mathsf{fn}(\Omega'_O) \subseteq \mathcal{N}'_O$ et $\mathsf{fn}(O'_i) \subseteq \mathcal{N}'_O$ pour tout i sont immédiatement vérifiées. Ce qui achève de vérifier les conditions pour le cas où M émet le message.

Enfin, la dernière possibilité à envisager est une communication où $\prod_i l_i[\![O_i]\!]$ émet le message. Dans ce cas les deux transitions qui composent la communication sont $\Omega_1 \rhd M \xrightarrow{(\Phi)c?V} \Omega'_1 \rhd M'$ et $\Omega_2 \rhd \prod_i l_i[\![O_i]\!] \xrightarrow{(\Phi)c!V} \Omega'_2 \rhd \prod_i l_i[\![O'_i]\!]$. De manière similaire au cas précédent, on peut conclure que l est un certain l_{i_0}, et que $\Omega_O \vdash_{l_{i_0}} c : \mathrm{W}\langle\mathsf{T}\rangle$. Par ailleurs, la structure très simple de $\prod_i l_i[\![O_i]\!]$ impose que Φ soit vide. Alors la transition de M peut aussi prendre place dans l'environnement Ω' où l_{i_0} est accessible : $\Omega' \rhd M \xrightarrow{c?V} \Omega' \rhd M'$. La bisimilarité de M et N implique par conséquent

$\Omega' \rhd N \xrightarrow{c?V} \Omega' \rhd N'$ avec $\Omega' \vDash M' \ {}_{\tilde{a}^i_M}{\approx}^{al}_{\tilde{a}^i_N} \ N'$. On conserve naturellement Ω_O inchangé et on voit immédiatement que toutes les hypothèses sont préservées, ce qui achève cette preuve.

Le raisonnement est totalement identique lorsque le système contenant N effectue l'action. □

Lemme 4.43 (Fermeture de la bisimulation par fermeture de portées)**.** *Si* $\Omega; a : \mathsf{E} \vDash M \ {}_{\tilde{a}_M}{\approx}^{al}_{\tilde{a}_N} \ N$ *et les configurations* $\Omega \rhd_{\tilde{a}_M} (\mathsf{new}\, a : \mathsf{E})\, M$ *et* $\Omega \rhd_{\tilde{a}_N} (\mathsf{new}\, a : \mathsf{E})\, N$ *sont bien formées, alors* $\Omega \vDash (\mathsf{new}\, a : \mathsf{E})\, M \ {}_{\tilde{a}_M}{\approx}^{al}_{\tilde{a}_N} \ (\mathsf{new}\, a : \mathsf{E})\, N$.

Démonstration. Considérons la relation $\mathcal{S}$

$$\left\{\begin{array}{l} \Omega \vDash (\mathsf{new}\, a : \mathsf{E})\, M \ {}_{\tilde{a}_M}\mathcal{S}_{\tilde{a}_N} \ (\mathsf{new}\, a : \mathsf{E})\, N \\ \Omega \vDash (\mathsf{new}\, a : \mathsf{E})\, M \ {}_{\tilde{a}_M}\mathcal{S}_{\tilde{a}_N, a} \ N \\ \Omega \vDash M \ {}_{\tilde{a}_M, a}\mathcal{S}_{\tilde{a}_N} \ (\mathsf{new}\, a : \mathsf{E})\, N \\ \Omega \vDash M \ {}_{\tilde{a}_M, a}\mathcal{S}_{\tilde{a}_N, a} \ N \end{array}\right.$$
$$\text{si} \quad \left\{\begin{array}{l} a \notin \tilde{a}_M \quad \text{et} \quad a \notin \tilde{a}_N \\ \Omega; a : \mathsf{E} \vDash M \ {}_{\tilde{a}_M}{\approx}^{al}_{\tilde{a}_N} \ N \\ \Omega \rhd_{\tilde{a}_M} (\mathsf{new}\, a : \mathsf{E})\, M \quad \text{et} \quad \Omega \rhd_{\tilde{a}_N} (\mathsf{new}\, a : \mathsf{E})\, N \quad \text{bien formées} \end{array}\right.$$

et la relation $\mathcal{R} = \approx^{al} \cup \mathcal{S}$. Montrons alors que $\mathcal{R}$ est une bisimulation loyale. La condition de définition des bisimulations loyales est directement vérifiée pour les quintuplets apparaissant dans la portion $\approx^{al}$ de $\mathcal{R}$.

Intéressons-nous donc à la partie $\mathcal{S}$, en particulier à une configuration annotée contenant M dans $\mathcal{S}$. Supposons que cette configuration annotée effectue une action μ. On identifie alors trois cas possibles :

1. μ est une action invisible ou une réception ;
2. μ est une émission révélant le nom a à l'observateur ;
3. μ est une émission passant le nom a sous silence.

action invisible ou réception Les deux possibilités, que la configuration annotée effectuant l'action soit $\Omega \rhd_{\tilde{a}_M} (\mathsf{new}\, a : \mathsf{E})\, M$ ou $\Omega \rhd_{\tilde{a}_M, a} M$, sont assez similaires. Pour le système $(\mathsf{new}\, a : \mathsf{E})\, M$, puisque l'action n'est pas une émission, la preuve de la transition correspondant à l'action utilise obligatoirement la règle (TE-C-NEW) pour $(\mathsf{new}\, a : \mathsf{E})$. Dans les deux cas, on extrait facilement de la preuve de la transition $\xrightarrow{\mu}$ une preuve de $\Omega \rhd M \xrightarrow{\mu} \Omega\, \mathsf{après}\, \mu \rhd M'$. Par affaiblissement, on en déduit $\Omega; a : \mathsf{E} \rhd M \xrightarrow{\mu} \Omega; a : \mathsf{E}\mathsf{après}\mu \rhd M'$. Naturellement l'action μ peut donc être effectuée par la configuration $\Omega; a : \mathsf{E} \rhd_{\tilde{a}_M} M$, et la bisimilarité avec N permet de conclure que la transition $\Omega; a : \mathsf{E} \rhd N \xRightarrow{\hat{\mu}} \Omega; a : \mathsf{E}\mathsf{après}\mu \rhd N'$ est prouvable, avec $\Omega\, \mathsf{après}\, \mu \vDash M' \ {}_{\tilde{a}_M}{\approx}^{al}_{\tilde{a}_N} \ N'$. Étant donné que l'action μ est activée dans Ω, cette transition peut être renforcée en $\Omega \rhd N \xRightarrow{\hat{\mu}} \Omega\, \mathsf{après}\, \mu \rhd N'$. Si la configuration contenant N est $\Omega \rhd_{\tilde{a}_N} (\mathsf{new}\, a : \mathsf{E})\, N$, une application de la règle (TE-C-NEW) permet de prouver la transition qui donne l'action faible $\Omega \rhd_{\tilde{a}_N} (\mathsf{new}\, a : \mathsf{E})\, N \xRightarrow{\hat{\mu}} \Omega\, \mathsf{après}\, \mu \rhd_{\tilde{a}_N} (\mathsf{new}\, a : \mathsf{E})\, N'$. S'il s'agit de la configuration $\Omega \rhd_{\tilde{a}_N, a} N$, l'action faible $\Omega \rhd_{\tilde{a}_N} N \xRightarrow{\hat{\mu}} \Omega\, \mathsf{après}\, \mu \rhd_{\tilde{a}_N, a} N'$ découle directement de la transition précédente.

Enfin, $\Omega; a : \mathsf{E}\mathsf{après}\mu$ doit être équivalent à $\Omega\mathsf{après}\mu; a : \mathsf{E}$ car $\Omega; a : \mathsf{E}\mathsf{après}\mu$ est soit $\Omega; a : \mathsf{E}$ soit $\Omega; a : \mathsf{E}; \Phi$, si $\mu = (\Phi)c?V$. En effet, dans ce cas, puisque μ est aussi activée dans l'environnement Ω, l'environnement Φ

ne peut faire référence au nom a, donc Φ et $a : \mathsf{E}$ commutent. Ce dernier constat permet de conclure que les deux configurations de $\mathcal{S}$ évoluent bien par l'action μ en deux nouvelles configurations de cette relation.

émission révélant a μ est dans ce cas de la forme $(\tilde{b}, a)c!V$, avec a apparaissant donc dans V. Supposons que la configuration de départ soit $\Omega \rhd_{\tilde{a}_M} (\mathsf{new}\, a : \mathsf{E})\, M$. L'action

$$\Omega \rhd_{\tilde{a}_M} (\mathsf{new}\, a : \mathsf{E})\, M \xrightarrow{(\tilde{b},a)c!V} \Omega,\langle V : \Omega^r(c)\rangle \circledast l \rhd_{\tilde{a}'_M} M'$$

est alors prouvée par une transition

$$\Omega \rhd (\mathsf{new}\, a : \mathsf{E})\, M \xrightarrow{(a:\mathsf{E};\Phi_M)c!V} \Omega,\langle V : \Omega^r(c)\rangle \circledast l \rhd M'$$

dont la preuve commence obligatoirement par une règle (TE-OUV) appliquée sur

$$\Omega \rhd M \xrightarrow{(\Phi_M)c!V} \Omega,\langle V : \Omega^r(c)\rangle \circledast l \rhd M'$$

Si la configuration de départ est en fait $\Omega \rhd_{\tilde{a}_M, a} M$, l'action ne peut être prouvée que par une telle transition.
Cette transition peut s'affaiblir en

$$\Omega; a : \mathsf{E} \rhd M \xrightarrow{(\Phi_M)c!V} (\Omega; a : \mathsf{E}),\langle V : (\Omega; a : \mathsf{E})^r(c)\rangle \circledast l \rhd M'$$

qui prouve l'action

$$\Omega; a : \mathsf{E} \rhd_{\tilde{a}_M} M \xrightarrow{(\tilde{b})c!V} (\Omega; a : \mathsf{E}),\langle V : (\Omega; a : \mathsf{E})^r(c)\rangle \circledast l \rhd_{\tilde{a}'_M} M'$$

donc l'action faible

$$\Omega; a : \mathsf{E} \rhd_{\tilde{a}_N} N \xRightarrow{(\tilde{b})c!V} (\Omega; a : \mathsf{E}),\langle V : (\Omega; a : \mathsf{E})^r(c)\rangle \circledast l \rhd_{\tilde{a}'_N} N'$$

avec

$$(\Omega; a : \mathsf{E}),\langle V : (\Omega; a : \mathsf{E})^r(c)\rangle \circledast l \vDash M' \; {}_{\tilde{a}'_M}{\approx}^{al}_{\tilde{a}'_N} \; N'$$

Cette action faible suppose une transition

$$\Omega; a : \mathsf{E} \rhd N \xRightarrow{(\Phi_N)c!V} (\Omega; a : \mathsf{E}),\langle V : (\Omega; a : \mathsf{E})^r(c)\rangle \circledast l \rhd N'$$

qui peut être renforcée en

$$\Omega \rhd N \xRightarrow{(\Phi_N)c!V} \Omega,\langle V : \Omega^r(c)\rangle \circledast l \rhd N'$$

d'où l'on déduit soit

$$\Omega \rhd_{\tilde{a}_N, a} N \xRightarrow{(\tilde{b},a)c!V} \Omega,\langle V : \Omega^r(c)\rangle \circledast l \rhd_{\tilde{a}'_N} N'$$

soit

$$\Omega \rhd_{\tilde{a}_N} (\mathsf{new}\, a : \mathsf{E})\, N \xRightarrow{(\tilde{b},a)c!V} \Omega,\langle V : \Omega^r(c)\rangle \circledast l \rhd_{\tilde{a}'_N} N'$$

Puisque $(\Omega; a : \mathsf{E}),\langle V : (\Omega; a : \mathsf{E})^r(c)\rangle \circledast l <: \Omega,\langle V : \Omega^r(c)\rangle \circledast l$, on conclut, par le corollaire 4.37, que

$$\Omega,\langle V : \Omega^r(c)\rangle \circledast l \vDash M' \; {}_{\tilde{a}'_M}{\approx}^{al}_{\tilde{a}'_N} \; N'$$

ce qui achève de prouver que ce quintuplet est dans $\mathcal{R}$.

émission ne révélant pas a Dans ce cas, μ est une action de la forme $(\tilde{b})c!V$ où V ne contient pas le nom a, donc a n'apparaît pas dans l'ensemble $\tilde{b}$. Que l'action soit effectuée par la configuration $\Omega \rhd_{\tilde{a}_M,a} M$ ou $\Omega \rhd_{\tilde{a}_M} (\mathsf{new}\, a : \mathsf{E})\, M$ (auquel cas les règles (TE-OUV) ou (TE-C-NEW) peuvent apparaître), la preuve de cette action passe par une transition

$$\Omega \rhd M \xrightarrow{(\Phi_M)c!V} \Omega, \langle V : \Omega^r(c)\rangle \circledcirc l \rhd M'$$

Par affaiblissement,

$$\Omega; a : \mathsf{E} \rhd M \xrightarrow{(\Phi_M)c!V} (\Omega; a : \mathsf{E}), \langle V : (\Omega; a : \mathsf{E})^r(c)\rangle \circledcirc l \rhd M'$$

ce qui prouve l'action

$$\Omega; a : \mathsf{E} \rhd_{\tilde{a}_M} M \xrightarrow{(\tilde{b})c!V} (\Omega; a : \mathsf{E}), \langle V : (\Omega; a : \mathsf{E})^r(c)\rangle \circledcirc l \rhd_{\tilde{a}'_M} M'$$

donc l'action faible

$$\Omega; a : \mathsf{E} \rhd_{\tilde{a}_N} N \xRightarrow{(\tilde{b})c!V} (\Omega; a : \mathsf{E}), \langle V : (\Omega; a : \mathsf{E})^r(c)\rangle \circledcirc l \rhd_{\tilde{a}'_N} N'$$

soit

$$\Omega; a : \mathsf{E} \rhd N \xRightarrow{(\Phi_N)c!V} (\Omega; a : \mathsf{E}), \langle V : (\Omega; a : \mathsf{E})^r(c)\rangle \circledcirc l \rhd N'$$

qui se renforce en

$$\Omega \rhd N \xRightarrow{(\Phi_N)c!V} \Omega, \langle V : \Omega^r(c)\rangle \circledcirc l \rhd N'$$

Les trois actions que l'on peut déduire de cette transition sont alors :

$$\Omega \rhd_{\tilde{a}_N} (\mathsf{new}\, a : \mathsf{E})\, N \xRightarrow{(\tilde{b})c!V} \Omega, \langle V : \Omega^r(c)\rangle \circledcirc l \rhd_{\tilde{a}'_N} (\mathsf{new}\, a : \mathsf{E})\, N'$$

$$\Omega \rhd_{\tilde{a}_N} (\mathsf{new}\, a : \mathsf{E})\, N \xRightarrow{(\tilde{b})c!V} \Omega, \langle V : \Omega^r(c)\rangle \circledcirc l \rhd_{\tilde{a}'_N,a} N'$$

et

$$\Omega \rhd_{\tilde{a}_N,a} N \xRightarrow{(\tilde{b})c!V} \Omega, \langle V : \Omega^r(c)\rangle \circledcirc l \rhd_{\tilde{a}'_N,a} N'$$

par une application des règles (TE-C-NEW) ou (TE-OUV) sur la transition, ou bien directement.

Enfin, puisque $a \neq c$ et que V ne mentionne pas a, l'environnement $(\Omega; a : \mathsf{E}), \langle V : (\Omega; a : \mathsf{E})^r(c)\rangle \circledcirc l$ est équivalent à $\Omega, \langle V : \Omega^r(c)\rangle \circledcirc l; a : \mathsf{E}$. On sait donc $\Omega, \langle V : \Omega^r(c)\rangle \circledcirc l; a : \mathsf{E} \vDash M' \; {}_{\tilde{a}'_M}\!\approx^{al}_{\tilde{a}'_N} N'$ qui prouve donc que les deux configurations annotées que l'on obtient dans tous les cas de figure sont effectivement dans la relation $\mathcal{S}$.

Le raisonnement est bien entendu symétrique dans le cas où l'on considère une action effectuée par la configuration contenant N. □

Ces trois lemmes permettent enfin de prouver que la bisimulation est effectivement incluse dans la congruence.

Lemme 4.44 (La bisimilarité loyale est incluse dans la congruence typée annotée barbue fermée par réduction et loyale)**.** $\approx^{al} \subseteq \cong^{al}$

Démonstration. La bisimilarité loyale est, par définition, une relation typée annotée symétrique. Les lemmes 4.41 et 4.43 et le théorème 4.42 prouvent qu'elle est loyalement contextuelle.

Prouvons donc qu'elle est fermée par réductions. Supposons $\Omega \vDash M \ {}_{\tilde{a}_M}\approx^{al}_{\tilde{a}_N} N$. Si $M \longrightarrow M'$ alors le théorème 4.25 donne l'existence d'un système M'' tel que $M'' \equiv M'$ et $\Omega \rhd_{\tilde{a}_M} M \xrightarrow{\tau} \Omega \rhd_{\tilde{a}_M} M''$. La bisimilarité implique alors que $\Omega \rhd_{\tilde{a}_N} N \stackrel{\hat{\tau}}{\Longrightarrow} \Omega \rhd_{\tilde{a}_N} N'$, avec $\Omega \vDash M'' \ {}_{\tilde{a}_M}\approx^{al}_{\tilde{a}_N} N'$. Le théorème 4.25 permet également de conclure que $N \Longrightarrow N'$. Enfin le corollaire 4.33 et la transitivité de la bisimilarité permet de conclure $\Omega \vDash M' \ {}_{\tilde{a}_M}\approx^{al}_{\tilde{a}_N} N'$.

Enfin, prouvons que la bisimilarité loyale respecte les barbes typées. On voit aisément que $\Omega \rhd_{\tilde{a}_M} M \Downarrow c$ implique directement l'existence d'un V et d'un Φ_M tels que $\Omega \rhd_{\tilde{a}_M} M \xrightarrow{(\Phi_M)c!V}$. La bisimilarité entre M et N donne alors immédiatement $\Omega \rhd_{\tilde{a}_N} N \xrightarrow{(\Phi_N)c!V}$. Par ailleurs, un système capable d'effectuer une action $(\Phi_N)c!V$ est nécessairement structurellement congru à $(\mathsf{new}\,\Phi)(N'' \mid l[\![c\,!\,\langle V\rangle\, P]\!])$ car les portées des noms de $\mathsf{dom}(\Phi)$ peuvent toujours être étendues à tout le système. Cette remarque permet de conclure que $\Omega \rhd_{\tilde{a}_N} N \Downarrow c$.

$\cong^{al}$ étant la plus grande relation typée vérifiant ces conditions, $\approx^{al} \subseteq \cong^{al}$. □

4.7.6 Réciproque

Penchons-nous maintenant sur la réciproque, l'inclusion de la congruence loyale dans la bisimilarité loyale. Pour prouver ce résultat construisons une relation typée contenant $\cong^{al}$ qui soit une bisimulation loyale : cette relation sera alors de facto incluse dans la plus grande, $\approx^{al}$.

Le choix de la relation typée utilisée dans cette démarche doit permettre de faciliter les deux preuves d'inclusion. On définit donc une relation très proche de la congruence barbue loyale, de sorte que la preuve de l'inclusion de $\cong^{al}$ dans cette relation soit immédiate.

Définition 4.45 (Relation typée annotée loyalement et parallèlement contextuelle)**.** *Une relation typée annotée $\mathcal{R}$ entre systèmes est dite* loyalement et parallèlement contextuelle *si et seulement si les conditions suivantes sont vérifiées.*

- *Si $\Omega \vDash M \ {}_{\tilde{a}_M}\mathcal{R}_{\tilde{a}_N} N$, et Ω' est une extension loyale de Ω telle que $\mathsf{dom}(\Omega')$ ne contienne que des noms inédits alors $\Omega;\Omega' \vDash M \ {}_{\tilde{a}_M}\mathcal{R}_{\tilde{a}_N} N$.*
- *Si $\Omega \vDash M \ {}_{\tilde{a}_M}\mathcal{R}_{\tilde{a}_N} N$, $k \in \mathcal{A}_\Omega$, $\Omega \vdash k[\![P]\!]$ et $\mathsf{fn}(P) \cap (\tilde{a}_M \cup \tilde{a}_N) = \emptyset$ alors $\Omega \vDash M \mid k[\![P]\!] \ {}_{\tilde{a}_M}\mathcal{R}_{\tilde{a}_N} N \mid k[\![P]\!]$.*

Définition 4.46 (Congruence parallèle typée annotée barbue fermée par réduction et loyale)**.** *On appelle* congruence parallèle typée annotée barbue fermée par réduction et loyale *la plus grande relation typée annotée symétrique loyalement et parallèlement contextuelle, fermée par réduction et qui respecte les barbes typées. On la note $\cong^p$.*

La définition de $\cong^p$ se résumant à une version affaiblie de celle de $\cong^{al}$, on voit immédiatement que $\cong^{al} \subseteq \cong^p$. La preuve de $\cong^p \subseteq \approx^{al}$ est plus complexe. En effet, pour prouver que $\cong^p$ est une bisimulation, il faut prouver que, pour tout $\Omega \vDash M \ {}_{\tilde{a}_M}\cong^p_{\tilde{a}_N} N$, toutes les actions de la configurations $\Omega \rhd_{\tilde{a}_M} M$ peuvent être mimées par $\Omega \rhd_{\tilde{a}_N} N$. Quand ces actions sont invisibles, la forte corrélation entre $\xrightarrow{\tau}$ et $\longrightarrow$ et le fait que $\cong^p$ soit fermée par réductions permettent de conclure. En revanche, les actions visibles sont plus complexes à traiter. Pour

cela, on rappelle l'intuition avancée pour définir les actions : on cherchait alors à identifier les contextes caractéristiques avec lesquels il est possible d'interagir. Puisque la relation $\cong^p$ est fermée par ajout de contexte de la forme $[\cdot] \mid k[\![P]\!]$, nous allons définir de tels contextes permettant d'identifier une action particulière du système placé dans le contexte.

Considérons une configuration $\Omega \rhd_{\tilde{a}_M} M$ capable d'effectuer l'émission $(\tilde{b})c!V$ et construisons pas à pas un contexte caractéristique de cette action. Puisque cette émission est visible dans Ω, on sait que la localité l dans lequel est situé le canal c est accessible pour Ω, donc la contextualité de $\cong^p$ autorisera l'utilisation d'un contexte de la forme $[\cdot] \mid l[\![P]\!]$. Qui plus est, on sait aussi que le jugement $\Omega \vdash_l c : \mathrm{R}\langle \mathsf{T} \rangle$ est prouvable. Le processus P pourra donc simplement commencer en effectuant une réception sur le canal $c : P = c\,?\,(X : \Omega^r(c))\,P'$. La continuation P' doit ensuite vérifier que la valeur reçue, X, correspond effectivement à celle attendue, à savoir V où les noms $\tilde{b}$ sont nouveaux. Quand l'action ne révèle aucun nouveau nom, c'est-à-dire $\tilde{b} = \emptyset$, cette vérification se résume à un simple test $V = X$, c'est-à-dire une simple cascade de tests selon la décomposition des égalités de valeurs décrite à la section 1.2. Mais si l'action est $(b)c!(b, b)$, X aura forcément la forme (x, y) et P' devra effectuer deux opérations différentes :

- vérifier que les variables x et y ont été égalisées par la communication ;
- vérifier que le nom par lequel elles sont substituées est authentiquement nouveau.

La première vérification à effectuer est simple. La seconde suppose en revanche de disposer de l'ensemble des noms qui ne sont pas nouveaux. Cette vérification sera donc paramétrée par un ensemble $\mathcal{N}$ contenant tous les noms déjà connus. Cet ensemble devra donc vérifier les équations suivantes :

- $\mathcal{N} \cap \tilde{a}_M = \emptyset$, car $\tilde{a}_M$ est un ensemble de noms privés du système M, même si leur portée syntaxique a dû être ouverte ;
- $(\mathsf{fn}(M) \setminus \tilde{a}_M) \subseteq \mathcal{N}$;
- $\mathsf{dom}(\Omega) \subseteq \mathcal{N}$.

En somme, une action $(a, b)c!(a, b, d, b)$ sera testée en exécutant une réception $c\,?\,((x_1, x_2, x_3, x_4))$ suivie des vérifications suivantes :

- $x_3 = d$,
- $x_1 \neq e$, pour tout $e \in \mathcal{N}$,
- $x_2 \neq e$, pour tout $e \in \mathcal{N}$,
- $x_2 = x_4$,
- et enfin $x_1 \neq x_2$.

On a arbitrairement pris x_2 comme représentant de b. On définit donc, de façon générale, le test de la façon suivante :

Notation 4.47. *Si :*

- $\tilde{b} = \{b_1, \ldots\}$,
- (V, X) *a pour décomposition arborescente simultanée* $(v_1, x_1), \ldots, (v_n, x_n)$,
- $(p_i)_i$ *est une suite d'indices telle que* $v_{p_i} \notin \tilde{b}$,
- *pour tout* j, $(p_{j;i})_i$ *est une suite d'indices telle que* $v_{p_{j;i}} = b_j$,
- *les suites p sont complètes, c'est-à-dire* $\bigcup_i \{p_i\} \cup \bigcup_{j,k} \{p_{j;k}\}$ *contient tous les indices de* 1 *à* n,
- $\mathcal{N}$ *est un ensemble fini de noms dont l'intersection avec* $\tilde{b}$ *est vide,*

alors $X =_{\mathcal{N}} (\tilde{b})V$ *dénotera la cascade de tests :*

- $x_{p_i} = v_{p_i}$ *pour tout* i,
- $x_{p_{j;1}} \neq a$ *pour tout* $a \in \mathcal{N}$ *et tout* j,

– $x_{p_{j;1}} = x_{p_{j;i}}$ *pour tout* $i > 1$ *et tout* j,
– $x_{p_{j;1}} \neq x_{p_{k;1}}$ *pour tous* j *et* k *tels que* $j \neq k$.

Nous avons ainsi tous les outils pour décrire un contexte apte à interagir avec un système effectuant une émission $(\tilde{b})c!V$ et à certifier qu'il s'agit bien là de cette action précise. On attendra une dernière opération de la part du contexte : rapporter les connaissances acquises au cours de la communication. En effet l'émission enrichit normalement la connaissance de l'observateur de Ω en $\Omega,\langle V : \Omega^r(c)\rangle_{@}l$. On utilisera pour obtenir ce résultat dans le cas d'une congruence barbue une technique standard : le contexte accumulera en effet ses connaissances sous forme d'une valeur qu'il mettra à disposition en la diffusant sur un canal particulier dans une localité particulière. Définissons donc comment les environnements peuvent être codés en valeur.

Définition 4.48 (Réification des environnements)**.** *À tout environnement* Γ *bien formé dont la forme normale naturelle est*

$$s_1 : \text{LOC}, \ldots, s_m : \text{LOC}, \quad u_1 : \tilde{t}^{\star}_{i_1} \mapsto t_{i_1}, \ldots, u_n : \tilde{t}^{\star}_{i_n} \mapsto t_{i_n}, \quad v_1 : \mathsf{C}_1{}_{@}s_{j_1}, \ldots, v_o : \mathsf{C}_o{}_{@}s_{j_o}$$

on associe la valeur

$$V_\Gamma = ((s_1, \ldots, s_m), (u_1, \ldots, u_n, v_1, \ldots, v_o))$$

et le type

$$\mathsf{T}_\Gamma = \sum x_1, \ldots, x_m : \tilde{\text{LOC}}.\ \tilde{x}^{\star}_{i_1} \mapsto x_{i_1}, \ldots, \tilde{x}^{\star}_{i_n} \mapsto x_{i_n}, \mathsf{C}_1{}_{@}x_{j_1}, \ldots, \mathsf{C}_o{}_{@}x_{j_o}$$

Proposition 4.49 (Validité de la réification)**.** *Quels que soient l'environnement bien formé* Γ *et la localité* s *définie dans* Γ, $\Gamma \vdash_s V_\Gamma : \mathsf{T}_\Gamma$.

Démonstration. Ce résultat découle immédiatement de l'équivalence entre Γ et sa forme normale naturelle. □

On remarque que le type T_Γ associé à un environnement Γ est clos. Il sera par conséquent possible de communiquer un environnement sous forme de valeur.

L'environnement enrichi que l'on voudra transmettre est $\Omega,\langle V : \Omega^r(c)\rangle_{@}l$. Mais il contient de nouveaux noms, les $\tilde{b}$. Au sein du contexte, ces nouveaux noms sont uniquement connus via le motif X. Dans la notation 4.47, on avait identifié en particulier la variable notée $x_{p_{j;1}}$ du motif X comme étant substituée par le nom b_j lorsque X est substitué par la valeur V. Par conséquent l'environnement $\Omega,\langle V : \Omega^r(c)\rangle_{@}l$ où les nouveaux noms b_j sont remplacés par la variable $x_{p_{j;1}}$ qui lui correspond peut s'écrire

$$\Omega,\langle V\{^{x_{p_{1;1}}}/_{b_1}\}\cdots\{^{x_{p_{n;1}}}/_{b_n}\} : \Omega^r(c)\rangle_{@}l$$

Définition 4.50 (Contexte associé à une émission)**.** *On appellera* contexte associé à l'émission $(\tilde{b})c!V$ *pour un observateur connaissant* Ω *et les noms* $\mathcal{N}$*, et on notera* $\mathfrak{C}^{\Omega}_{\mathcal{N}}((\tilde{b})c!V)$*, le système :*

$$l[\![c\,?\,(X : \Omega^r(c)) \quad \textit{if } X =_{\mathcal{N}} (\tilde{b})V \textit{ then goto}_\pi\ \lambda.\ \omega\,!\,\langle V_{\Omega,\langle V\{^{x_{p_{1;1}}}/_{b_1}\}\cdots\{^{x_{p_{n;1}}}/_{b_n}\}:\Omega^r(c)\rangle_{@}l}\rangle \textit{ else stop}]\!]$$

si λ, π *et* ω *sont des noms inédits.*

On utilise dans cet environnement les lettres grecques λ, π et ω pour insister que ces noms sont inédits et introduits par l'observateur pour ce contexte. Afin d'alléger les notations au cours des preuves qui suivent, on notera $\Omega\lambda\pi\omega$ l'environnement $\Omega; \lambda : \mathsf{T}_1, \pi : \mathsf{T}_2, \omega : \mathsf{T}_3$. Les types précis associés aux différentes lettres ajoutées seront clairs à partir du contexte.

Lemme 4.51 (Légitimité du contexte associé à une émission)**.** *Soient un environnement Ω, une émission $(\tilde{b})c!V$ activée dans Ω, et un ensemble de noms $\mathcal{N}$. Alors* $\Omega; \lambda : \textsc{loc}; \pi : \star \mapsto \lambda; \omega : \textsc{rw}\langle \mathsf{T}_{\Omega \mathsf{après}(\tilde{b})c!V} \rangle_{@} \lambda \vdash \mathfrak{C}^{\Omega}_{\mathcal{N}}((\tilde{b})c!V)$.

Démonstration. Le cœur de cette preuve est le fait que

$$\Omega\lambda\pi\omega \vdash_{\lambda} \omega\,!\left\langle V_{\Omega,\langle V\{^{x_{p_{1;1}}}/_{b_1}\}\cdots\{^{x_{p_{n;1}}}/_{b_n}\}:\Omega^r(c)\rangle @ l}\right\rangle$$

Pour obtenir ce jugement, commençons par constater que

$$[\Omega\lambda\pi\omega;\langle X : \Omega^r(c)\rangle_{@} l]_{X=\mathcal{N}(\tilde{b})V} <: \Omega,\langle V\{^{x_{p_{1;1}}}/_{b_1}\}\cdots\{^{x_{p_{n;1}}}/_{b_n}\} : \Omega^r(c)\rangle_{@} l$$

en considérant une hypothèse de l'environnement de droite :

- si elle est dans Ω elle doit aussi être dans $[\Omega\lambda\pi\omega;\langle X : \Omega^r(c)\rangle_{@} l]_{X=\mathcal{N}(\tilde{b})V}$;
- si elle porte sur un $x_{p_1^{b_i}}$, elle doit apparaître concernant un $x_{p_j^{b_i}}$ dans l'environnement de gauche ; l'extension substitutive transpose alors nécessairement cette hypothèse pour $x_{p_1^{b_i}}$ d'après la notation 4.47 ;
- sinon, elle doit porter sur un nom de V n'appartenant pas à $\tilde{b}$, donc un certain v_{p_i}, toujours selon la notation 4.47 ; alors l'hypothèse équivalente concernant x_{p_i} est transposée pour v_{p_i} par l'extension substitutive.

Le même raisonnement permet de transposer les identifiants apparaissant dans les types de ces hypothèses, ce qui permet de conclure au sous-typage entre les deux environnements.

Par ailleurs,

$$\mathsf{T}_{\Omega,\langle V\{^{x_{p_{1;1}}}/_{b_1}\}\cdots\{^{x_{p_{n;1}}}/_{b_n}\}:\Omega^r(c)\rangle @ l} = \mathsf{T}_{\Omega,\langle V:\Omega^r(c)\rangle @ l}$$

car les variables x_i et les noms b_i sont inédits dans Ω.

La propriété 4.49 de validité de la réification permet alors de conclure. □

Vérifions maintenant que le contexte que nous venons de construire remplit effectivement son office, c'est-à-dire qu'il caractérise l'action $\xrightarrow{(\tilde{b})c!V}$. Cette vérification doit porter sur les deux directions : composé avec un système effectuant cette action, le contexte devra se réduire totalement, mais cette réduction totale devra aussi permettre de déduire que le système a effectué l'action. Dans le cadre de la congruence $\cong^p$, il est indispensable que l'observateur puisse distinguer quand la réduction est totale afin d'en tirer cette conclusion. Or les seuls éléments observables sont les barbes et

$$\Omega \rhd_{\tilde{a}_M} M \mid \mathfrak{C}^{\Omega}_{\mathcal{N}}((\tilde{b})c!V) \Downarrow \omega$$

dès que la configuration $\Omega \rhd_{\tilde{a}_M} M$ *peut* effectuer l'action $(\tilde{b})c!V$, pas uniquement lorsque cette action a été effectuée. Afin de détecter précisément si la réduction du contexte est vraiment totale, rajoutons donc une barbe supplémentaire, ne disparaissant que lorsque le calcul est terminé grâce au petit contexte suivant :

$$\mathfrak{C}_{fin} = \lambda[\![\delta\,!\,\langle\rangle \mid \omega\,?\,(X : \mathsf{T})\,\delta\,?\,()\,\omega\,!\,\langle X\rangle]\!]$$

Lemme 4.52 (Définissabilité des émissions)**.** *Soient un environnement Ω, une émission $\alpha = (\tilde{b})c!V$ activée dans cet environnement, un système M et un ensemble de noms $\mathcal{N}$ tel que $\tilde{a}_M \cap \mathcal{N} = \emptyset$ et $(\mathsf{fn}(M) \setminus \tilde{a}_M) \subseteq \mathcal{N}$.*

$$\Omega \rhd_{\tilde{a}_M} M \xrightarrow{\alpha} \Omega \text{ après } \alpha \rhd_{\tilde{a}'_M} M'$$

implique

$$M \mid \mathfrak{C}^{\Omega}_{\mathcal{N}}(\alpha) \mid \mathfrak{C}_{fin} \Longrightarrow (\mathit{new}\,\Phi)(M' \mid \mathfrak{C}_d \mid \lambda[\![\omega\,!\,\langle V_{\Omega \text{après} \alpha}\rangle]\!])$$

où $\mathfrak{C}_d = \lambda[\![\mathit{stop}]\!] \mid \lambda[\![\mathit{stop}]\!]$ et $\mathsf{dom}(\Phi) = (\tilde{b} \cup \tilde{a}'_M) \setminus \tilde{a}_M$.

Réciproquement, si

$$M \mid \mathfrak{C}^{\Omega}_{\mathcal{N}}(\alpha) \mid \mathfrak{C}_{fin} \Longrightarrow M''$$

avec $\Omega\lambda\pi\omega\delta \rhd_{\tilde{a}_M} M'' \Downarrow \omega$ et $\Omega\lambda\pi\omega\delta \rhd_{\tilde{a}_M} M'' \not\Downarrow \delta$ où $\Omega\lambda\pi\omega\delta = \Omega; \lambda : \text{LOC}; \pi : \star \mapsto \lambda; \omega : \text{RW}\langle \mathsf{T}_{\Omega \text{après} \alpha}\rangle_{@}\lambda; \delta : \text{RW}\langle\rangle_{@}\lambda$ alors M'' doit être équivalent au système $(\mathit{new}\,\Phi)(M' \mid \mathfrak{C}_d \mid \lambda[\![\omega\,!\,\langle V_{\Omega \text{après} \alpha}\rangle]\!])$ avec

$$\Omega \rhd_{\tilde{a}_M} M \stackrel{\alpha}{\Longrightarrow} \Omega \text{ après } \alpha \rhd_{\tilde{a}'_M} M'$$

$\tilde{b} = (\mathsf{dom}(\Phi) \cup \tilde{a}_M) \cap \mathsf{fn}(V)$ et $\tilde{a}'_M = (\mathsf{dom}(\Phi) \cup \tilde{a}_M) \setminus \mathsf{fn}(V)$.

Démonstration. Une action

$$\Omega \rhd_{\tilde{a}_M} M \xrightarrow{\alpha} \Omega \text{ après } \alpha \rhd_{\tilde{a}'_M} M'$$

implique une transition

$$\Omega \rhd M \xrightarrow{(\Phi)c!V} \Omega \text{ après } \alpha \rhd M'$$

avec $\mathsf{dom}(\Phi) = (\tilde{b} \cup \tilde{a}'_M) \setminus \tilde{a}_M$. Par ailleurs on voit aisément que

$$\Omega\lambda\pi\omega\delta \rhd \mathfrak{C}^{\Omega}_{\mathcal{N}}(\alpha) \xrightarrow{(\Phi)c?V} \Omega\lambda\pi\omega\delta \text{ après } (\Phi)c?V \rhd \mathfrak{C}^{\Omega}_{\mathcal{N}}(\alpha)'$$

La règle (TE-COMM) permet donc d'en conclure que

$$\Omega\lambda\pi\omega\delta \rhd M \mid \mathfrak{C}^{\Omega}_{\mathcal{N}}(\alpha) \xrightarrow{\tau} \Omega\lambda\pi\omega\delta \rhd (\mathsf{new}\,\Phi)(M \mid \mathfrak{C}^{\Omega}_{\mathcal{N}}(\alpha)')$$

La substitution $\{^{V}/_{X}\}$ appliquée à la continuation de la réception dans le contexte $\mathfrak{C}^{\Omega}_{\mathcal{N}}(\alpha)$ donne à $\mathfrak{C}^{\Omega}_{\mathcal{N}}(\alpha)'$ la forme

$$l[\![\mathsf{if}\ V =_{\mathcal{N}} (\tilde{b})V\ \mathsf{then}\ \mathsf{goto}_{\pi}\ \lambda.\ \omega\,!\,\langle V_{\Omega,(V\{^{v_{p_{1;1}}}/_{b_1}\}\cdots\{^{v_{p_{n;1}}}/_{b_n}\}:\Omega^r(c))_{@}l}\rangle\ \mathsf{else}\ \mathsf{stop}]\!]$$

Le test effectué correspond à la cascade décrite dans la notation 4.47. En substituant les x_i par les v_i qui leur sont associés, ces tests sont alors les suivants :

1. $v_{p_i} = v_{p_i}$ pour tout i,
2. $v_{p_{j;1}} \neq a$ pour tout $a \in \mathcal{N}$ et tout j, avec, par définition de $p_1^{b_j}$, $v_{p_{j;1}} = b_j$,
3. $v_{p_{j;1}} = v_{p_{j;i}}$ pour tout $i > 1$ et tout j, soit, encore par définition de $p_{j;i}$, $b_j = b_j$.
4. $v_{p_{j;1}} = v_{p_{k;1}}$ pour tous j et k avec $j \neq k$, soit, encore par définition de $p_{j;i}$ et $p_{k;i}$, $b_j \neq b_k$.

Les première et troisième séries de tests seront trivialement vraies. La quatrième le sera également par définition même des b_i comme les éléments de l'ensemble $\tilde{b}$. Le résultat pour la deuxième découle du fait que $b_j \in \tilde{a}_M \cup \mathsf{dom}(\Phi)$ où $\tilde{a}_M \cap \mathcal{N} = \emptyset$ par définition de $\mathcal{N}$ et $\mathsf{dom}(\Phi) \cap \mathcal{N} = \emptyset$ suivant la condition de Barendregt que les noms liés dans M, notamment ceux de Φ, sont uniques. On déduit alors facilement des égalités $b_i = v_{p_{i;1}}$ que

$$\Omega\lambda\pi\omega\delta \rhd (\mathsf{new}\,\Phi)(M \mid \mathfrak{C}^{\Omega}_{\mathcal{N}}(\alpha)') \stackrel{\tau}{\Longrightarrow} \Omega\lambda\pi\omega\delta \rhd (\mathsf{new}\,\Phi)(M \mid \lambda[\![\omega\,!\,\langle V_{\Omega,\langle V:\Omega^r(c)\rangle@l}\rangle]\!])$$

Les deux communications suivantes, celle sur ω entre le résidu de $\mathfrak{C}^{\Omega}_{\mathcal{N}}(\alpha)$ et $\mathfrak{C}_{fin}$ et celle interne à $\mathfrak{C}_{fin}$ sur δ, achèvent de prouver que :

$$\begin{array}{l}\Omega\lambda\pi\omega\delta \rhd M \mid \mathfrak{C}^{\Omega}_{\mathcal{N}}(\alpha) \mid \mathfrak{C}_{fin} \stackrel{\tau}{\Longrightarrow} \\ \quad \Omega\lambda\pi\omega\delta \rhd (\mathsf{new}\,\Phi_1)((\mathsf{new}\,\Phi_2)(M \mid \lambda[\![\mathsf{stop}]\!]) \mid \lambda[\![\mathsf{stop}]\!] \mid \lambda[\![\omega\,!\,\langle V_{\Omega\mathsf{après}\alpha}\rangle]\!])\end{array}$$

avec $\Phi \equiv \Phi_1;\Phi_2$, d'où le théorème 4.25 déduit

$$M \mid \mathfrak{C}^{\Omega}_{\mathcal{N}}(\alpha) \mid \mathfrak{C}_{fin} \Longrightarrow (\mathsf{new}\,\Phi)(M \mid \lambda[\![\mathsf{stop}]\!] \mid \lambda[\![\mathsf{stop}]\!] \mid \lambda[\![\omega\,!\,\langle V_{\Omega,\langle V:\Omega^r(c)\rangle@l}\rangle]\!])$$

Prouvons maintenant la réciproque.

$$M \mid \mathfrak{C}^{\Omega}_{\mathcal{N}}(\alpha) \mid \mathfrak{C}_{fin} \Longrightarrow M''$$

implique, toujours par le théorème 4.25 :

$$\Omega\lambda\pi\omega\delta \rhd M \mid \mathfrak{C}^{\Omega}_{\mathcal{N}}(\alpha) \mid \mathfrak{C}_{fin} \stackrel{\tau}{\longrightarrow}{}^{*}\ \Omega\lambda\pi\omega\delta \rhd M'''$$

où $M''' \equiv M''$ et $\Omega\lambda\pi\omega\delta \rhd_{\tilde{a}_M} M''' \not\Downarrow \delta$. Or le canal δ n'apparaît que dans le contexte $\mathfrak{C}_{fin}$ qui exhibe une barbe sauf si les préfixes $\delta\,?\,()$ et $\delta\,!\,\langle\rangle$ ont été consommés par une communication, ce qui impose que le préfixe $\omega\,?\,(X:\mathsf{T})$ de $\mathfrak{C}_{fin}$ ait aussi été consommé. Pour la même raison, la seule étape qui peut l'avoir consommé est une communication avec l'émission sur ω dans le contexte $\mathfrak{C}^{\Omega}_{\mathcal{N}}(\alpha)$. Enfin, en remontant plus loin dans l'histoire des transitions $\stackrel{\tau}{\longrightarrow}{}^{*}$, le test $X =_{\mathcal{N}} (\tilde{b})V$ a dû être un succès et le préfixe $c\,?\,(X:\Omega^r(c))$ être consommé, et ce par une communication avec M. La transition $\stackrel{\tau}{\longrightarrow}{}^{*}$ est donc nécessairement décomposable ainsi :

$$\begin{array}{ll}\Omega\lambda\pi\omega\delta \rhd M \mid \mathfrak{C}^{\Omega}_{\mathcal{N}}(\alpha) \mid \mathfrak{C}_{fin} & \\ \quad \stackrel{\tau}{\longrightarrow}{}^{*} & \Omega\lambda\pi\omega\delta \rhd M_0 \mid \mathfrak{C}^{\Omega}_{\mathcal{N}}(\alpha) \mid \mathfrak{C}_{fin} \\ \quad \stackrel{\tau}{\longrightarrow} & \Omega\lambda\pi\omega\delta \rhd (\mathsf{new}\,\Phi)(M_1 \mid \mathfrak{C}^{\Omega}_{\mathcal{N}}(\alpha)') \mid \mathfrak{C}_{fin} \\ \quad \stackrel{\tau}{\longrightarrow}{}^{*} & \Omega\lambda\pi\omega\delta \rhd (\mathsf{new}\,\Phi_2)((\mathsf{new}\,\Phi_1)(M' \mid \lambda[\![\mathsf{stop}]\!]) \mid \lambda[\![\mathsf{stop}]\!] \mid \lambda[\![\omega\,!\,\langle V''\rangle]\!])\end{array}$$

où $\Phi \equiv \Phi_2;\Phi_1$ et où la transition

$$\Omega\lambda\pi\omega\delta \rhd M_0 \mid \mathfrak{C}^{\Omega}_{\mathcal{N}}(\alpha) \mid \mathfrak{C}_{fin} \stackrel{\tau}{\longrightarrow} \Omega\lambda\pi\omega\delta \rhd (\mathsf{new}\,\Phi)(M_1 \mid \mathfrak{C}^{\Omega}_{\mathcal{N}}(\alpha)') \mid \mathfrak{C}_{fin}$$

est prouvée par (TE-COMM) avec les hypothèses

$$\Omega_{M_0} \rhd M_0 \stackrel{(\Phi)c!V'}{\longrightarrow} \Omega'_{M_0} \rhd M_1$$

et

$$\Omega_{\mathfrak{C}} \rhd \mathfrak{C}^{\Omega}_{\mathcal{N}}(\alpha) \stackrel{(\Phi)c?V'}{\longrightarrow} \Omega'_{\mathfrak{C}} \rhd \mathfrak{C}^{\Omega}_{\mathcal{N}}(\alpha)'$$

D'après le lemme 4.35 de renforcement, la transition

$$\Omega \rhd M_0 \xrightarrow{(\Phi)c!V'} \Omega;\langle V' : \Omega^r(c)\rangle \circledcirc l \rhd M_1$$

est prouvable, ce qui entraîne simplement

$$\Omega \rhd_{\tilde{a}_M} M \xrightarrow{(\tilde{b}')c!V'} \Omega;\langle V' : \Omega^r(c)\rangle \circledcirc l \rhd_{\tilde{a}'_M} M'$$

avec $\tilde{b}' = (\tilde{a}_M \cup \mathsf{dom}(\Phi)) \cap \mathsf{fn}(V')$ et $\tilde{a}'_M = (\tilde{a}_M \cup \mathsf{dom}(\Phi)) \setminus \mathsf{fn}(V')$. Il nous reste à vérifier que les actions $(\tilde{b}')c!V'$ et $(\tilde{b})c!V$ sont indistinguables, c'est-à-dire $V = V'$ et $b = b'$. Par définition, le test $V' =_{\mathcal{N}} (\tilde{b})V$, qui est nécessairement vérifié au cours des transitions $\xrightarrow{\tau}{}^*$, garantit les égalités suivantes :

- $v'_{p_i} = v_{p_i}$ pour tout i,
- $v'_{p_{j;1}} \neq a$ pour tout $a \in \mathcal{N}$ et tout j,
- $v'_{p_{j;1}} = v'_{p_{j;i}}$ pour tout $i > 1$ et tout j,
- $v'_{p_{j;1}} \neq v'_{p_{k;1}}$ pour tous j et k avec $j \neq k$.

Puisque les identifiants reconnaissables, v'_{p_i}, ont les valeurs attendues, et que les autres apparaissent aux positions idoines grâce aux trois dernières séries d'égalités, les deux actions sont égales à un éventuel renommage sur les noms liés.

Cette égalité permet de conclure que

$$\Omega \rhd_{\tilde{a}_M} M \xrightarrow{(\tilde{b})c!V} \Omega;\langle V' : \Omega^r(c)\rangle \circledcirc l \rhd_{\tilde{a}'_M} M'$$

et par conséquent que la valeur V'' transmise sur le canal ω dans M''' est $V_{\Omega\mathsf{après}(\tilde{b})c!V}$, d'après le même raisonnement qu'au début de cette preuve, ce qui en achève la seconde partie. □

Maintenant que nous avons vu comment définir un contexte pour une émission, regardons comment faire de même dans le cas d'une réception. Dans le cas où cette réception est simplement $c?V$, il suffit bien entendu de choisir le contexte $l[\![c\,!\,\langle V\rangle\,\omega\,!\,\langle V_\Omega\rangle]\!]$. Dans le cas d'une action $(\Phi)c?V$ où les noms de Φ doivent être générés, comme nous l'avons déjà remarqué, les règles (TE-AFF) du SLTE qui apparaissent dans la preuve de cette transition peuvent commuter avec les règles (TE-C-NEW) et (TE-C-PAR). Par conséquent, on se contentera de caractériser l'action $c?V$ par contexte.

Définition 4.53 (Contexte associé à une réception)**.** *On appellera* contexte associé à la réception $c?V$ *pour un observateur connaissant* Ω*, et on notera* $\mathfrak{C}(c?V)$*, le système :*

$$l[\![c\,!\,\langle V\rangle\,\mathit{goto}_\pi\,\lambda.\,\omega\,!\,\langle V_\Omega\rangle]\!]$$

si λ*,* π *et* ω *sont des noms inédits.*

Lemme 4.54 (Légitimité du contexte associé à une réception)**.** *Soient un environnement* Ω*, une réception* $c?V$ *activée dans* Ω*. Alors le contexte* $\mathfrak{C}(c?V)$ *est bien typé dans l'environnement* $\Omega; \lambda : \textsc{loc}; \pi : \star \mapsto \lambda; \omega : \textsc{rw}\langle \mathsf{T}_\Omega\rangle \circledcirc \lambda$*.*

Démonstration. Ce résultat découle immédiatement du fait que l'action est activée dans Ω et de la validité de la réification de l'environnement (proposition 4.49). □

Suivant la même démarche que pour les émissions, vérifions maintenant que ce contexte caractérise la réception.

Lemme 4.55 (Définissabilité des réceptions)**.** *Soient un environnement Ω, une émission $\alpha = c?V$ activée dans cet environnement et un système M.*

$$\Omega \rhd_{\tilde{a}_M} M \xrightarrow{\alpha} \Omega \rhd_{\tilde{a}_M} M'$$

implique

$$M \mid \mathfrak{C}(\alpha) \mid \mathfrak{C}_{fin} \Longrightarrow M' \mid \mathfrak{C}_d \mid \lambda[\![\omega\,!\,\langle V_\Omega\rangle]\!]$$

où $\mathfrak{C}_d = \lambda[\![\mathsf{stop}]\!] \mid \lambda[\![\mathsf{stop}]\!]$.

Réciproquement, si

$$M \mid \mathfrak{C}(\alpha) \mid \mathfrak{C}_{fin} \Longrightarrow M''$$

avec $\Omega\lambda\pi\omega\delta \rhd_{\tilde{a}_M} M'' \Downarrow \omega$ *et* $\Omega\lambda\pi\omega\delta \rhd_{\tilde{a}_M} M'' \not\Downarrow \delta$ *où* $\Omega\lambda\pi\omega\delta = \Omega; \lambda : \text{LOC}; \pi : \star \mapsto \lambda; \omega : \text{RW}\langle \mathsf{T}_{\Omega\text{après}\alpha}\rangle_@\lambda; \delta : \text{RW}\langle\rangle_@\lambda$ *alors M'' doit être équivalent au système $M' \mid \mathfrak{C}_d \mid \lambda[\![\omega\,!\,\langle V_\Omega\rangle]\!]$ avec*

$$\Omega \rhd_{\tilde{a}_M} M \stackrel{\alpha}{\Longrightarrow} \Omega \rhd_{\tilde{a}_M} M'$$

Démonstration. La première partie est évidente, il suffit d'y combiner la réception du système M avec l'émission que peut effectuer $\mathfrak{C}(\alpha)$. Les communications suivantes sur les canaux ω et δ ne posent pas plus de difficultés.

Pour la réciproque, la coïncidence des sémantiques assure l'existence d'une transition

$$\Omega\lambda\pi\omega\delta \rhd M \mid \mathfrak{C}(\alpha) \mid \mathfrak{C}_{fin} \xrightarrow{\tau}{}^* \Omega\lambda\pi\omega\delta \rhd M'''$$

Le raisonnement est alors identique que dans la preuve du lemme 4.52 : l'absence de barbe sur δ dans M'' garantit en cascade que les communications sur δ, ω et c ont eu lieu. On en déduit que, M''' doit être de la forme $M' \mid \mathfrak{C}_d \mid \lambda[\![\omega\,!\,\langle V_\Omega\rangle]\!]$ avec, dans l'environnement Ω où cette réception est activée, $\Omega \rhd M \stackrel{\alpha}{\Longrightarrow} \Omega \rhd M'$. □

Nous avons maintenant vu comment un contexte peut caractériser une action particulière d'un système, et comment il peut coder sous forme d'une valeur ses « connaissances » après cette action. Cependant, afin de prouver la coïncidence de la congruence avec la bisimulation, on veut pouvoir conclure, après ces réductions des contextes ajoutés, que les configurations annotées où la connaissance de l'environnement est enrichie sont toujours dans la congruence.

Lemme 4.56 (Ouverture des portées)**.** *Si*

$$\lambda : \text{LOC}, \pi : \star \mapsto \lambda, \omega : \text{RW}\langle \mathsf{T}_\Omega\rangle_@\lambda \vDash (\mathsf{new}\,\Phi_M)(M \mid \lambda[\![\omega\,!\,\langle V_\Omega\rangle]\!]) \;_{\tilde{a}_M}\!\cong^p_{\tilde{a}_N} (\mathsf{new}\,\Phi_N)(N \mid \lambda[\![\omega\,!\,\langle V_\Omega\rangle]\!])$$

et que les noms λ, π et ω n'apparaissent pas dans M ni N, alors

$$\Omega \vDash M \;_{\tilde{a}'_M}\!\cong^p_{\tilde{a}'_N} N$$

où $\tilde{a}'_M = (\tilde{a}_M \cup \mathsf{dom}(\Phi_M)) \setminus \mathsf{dom}(\Omega)$ *et* $\tilde{a}'_N = (\tilde{a}_N \cup \mathsf{dom}(\Phi_N)) \setminus \mathsf{dom}(\Omega)$.

Démonstration. Vérifions donc que la relation $\mathcal{R}$, définie par

$$\Omega \vDash M \;_{\tilde{a}'_M}\mathcal{R}_{\tilde{a}'_N} N$$

en conservant toutes les notations de l'énoncé du lemme, respecte toutes les conditions de définition de $\cong^p$ et est par conséquent incluse dans celle-ci.

typée La bonne formation de $\lambda\pi\omega \rhd_{\tilde{a}_M} (\mathsf{new}\,\Phi_M)(M \mid \lambda[\![\omega\,!\,\langle V_\Omega\rangle]\!])$ assure l'existence d'un environnement $\Gamma <: \lambda\pi\omega$ tel que $\Gamma; \Phi_M \vdash M$ et $\Gamma; \Phi_M \vdash V_\Omega : \mathsf{T}_\Omega$ d'où l'on peut déduire $\Gamma; \Phi_M <: \Omega$, ce qui prouve la bonne formation de la configuration $\Omega \rhd_{\tilde{a}'_M} M$. Le même raisonnement à propos de N garantit que $\mathcal{R}$ est une relation typée.

symétrique $\mathcal{R}$ est symétrique car elle hérite naturellement de la symétrie de $\cong^p$.

fermée par réduction Si M se réduit en M' en une étape, il va de soi que $(\mathsf{new}\,\Phi_M)(M \mid \lambda[\![\omega\,!\,\langle V_\Omega\rangle]\!])$ peut se réduire en $(\mathsf{new}\,\Phi_M)(M' \mid \lambda[\![\omega\,!\,\langle V_\Omega\rangle]\!])$. On en déduit que $(\mathsf{new}\,\Phi_N)(N \mid \lambda[\![\omega\,!\,\langle V_\Omega\rangle]\!])$ peut se réduire en N'' pour rester dans l'équivalence. Puisque λ et ω ne sont pas libres dans N, il est impossible que la composante $\lambda[\![\omega\,!\,\langle V_\Omega\rangle]\!]$ de ce système interagisse au cours de ces réductions. On sait donc

$$(\mathsf{new}\,\Phi_N)(N \mid \lambda[\![\omega\,!\,\langle V_\Omega\rangle]\!]) \Longrightarrow N'' \equiv (\mathsf{new}\,\Phi_N)(N' \mid \lambda[\![\omega\,!\,\langle V_\Omega\rangle]\!])$$

$\equiv \subseteq \cong^p$ et la transitivité de $\cong^p$ permettent de conclure

$$\lambda, \pi, \omega \vDash (\mathsf{new}\,\Phi_M)(M' \mid \lambda[\![\omega\,!\,\langle V_\Omega\rangle]\!])\ {}_{\tilde{a}_M}{\cong^p}_{\tilde{a}_N}\ (\mathsf{new}\,\Phi_N)(N' \mid \lambda[\![\omega\,!\,\langle V_\Omega\rangle]\!])$$

ce qui prouve que $\mathcal{R}$ est fermée par réductions.

respecte les barbes Supposons $\Omega \rhd_{\tilde{a}'_M} M \Downarrow a$. On sait alors qu'il existe l et p tels que $\Omega \vdash a : \mathrm{R}\langle\mathsf{T}\rangle_{@}l$ et $\Omega \vdash p : \star \mapsto l$. On définit le petit contexte $\mathfrak{C}_a$ par

$$\mathfrak{C}_a = \lambda[\![\omega\,?\,(X : \mathsf{T}_\Omega)\,\mathsf{goto}_{x_p}\,x_l.\ x_a\,?\,(Y : \Omega^r(a))\,\mathsf{goto}_\pi\,\lambda.\ \delta\,!\,\langle\rangle]\!]$$

où x_p, x_l et x_a sont les variables du motif X qui correspondent aux noms p, l et a dans la valeur V_Ω, et où δ est un nom inédit. Il entraîne les réductions suivantes :

$$\begin{aligned} M \mid \lambda[\![\omega\,!\,\langle V_\Omega\rangle]\!] \mid \lambda[\![\omega\,?\,(X : \mathsf{T}_\Omega)\,\mathsf{goto}_{x_p}\,x_l.\ x_a\,?\,(Y : \Omega^r(a))\,\mathsf{goto}_\pi\,\lambda.\ \delta\,!\,\langle\rangle]\!] \\ \Longrightarrow\equiv\quad ((\mathsf{new}\,\Phi)(M' \mid l[\![a\,!\,\langle V\rangle\ P]\!])) \mid \lambda[\![\mathsf{stop}]\!] \\ \qquad\qquad\qquad \mid \lambda[\![\mathsf{goto}_p\,l.\ a\,?\,(Y : \Omega^r(a))\,\mathsf{goto}_\pi\,\lambda.\ \delta\,!\,\langle\rangle]\!] \\ \Longrightarrow\quad ((\mathsf{new}\,\Phi)(M' \mid l[\![P]\!])) \mid \lambda[\![\mathsf{stop}]\!] \mid \lambda[\![\delta\,!\,\langle\rangle]\!] \end{aligned}$$

Bien entendu, ces réductions seraient bloquées au cas où M était incapable d'exhiber une barbe sur le canal a. Par conséquent ce contexte caractérise les barbes sur a au sens où $\Omega \rhd_{\tilde{a}'_M} M \Downarrow a$ si et seulement si $\lambda\pi\omega\delta \rhd_{\tilde{a}_M} \mathfrak{C}_a \mid (\mathsf{new}\,\Phi_M)\,M \mid \lambda[\![\omega\,!\,\langle V_\Omega\rangle]\!] \Downarrow \delta$. Or par fermeture par extension loyale (pour le nouveau canal δ) et par contexte parallèle de $\cong^p$, on sait

$$\lambda\pi\omega\delta \vDash \mathfrak{C}_a \mid (\mathsf{new}\,\Phi_M)\,M \mid \lambda[\![\omega\,!\,\langle V_\Omega\rangle]\!]\ {}_{\tilde{a}_M}{\cong^p}_{\tilde{a}_N}\ \mathfrak{C}_a \mid (\mathsf{new}\,\Phi_N)\,N \mid \lambda[\![\omega\,!\,\langle V_\Omega\rangle]\!]$$

d'où l'on peut déduire que $\lambda\pi\omega\delta \rhd_{\tilde{a}_M} \mathfrak{C}_a \mid (\mathsf{new}\,\Phi_M)\,M \mid \lambda[\![\omega\,!\,\langle V_\Omega\rangle]\!] \Downarrow \delta$ si et seulement si $\lambda\pi\omega\delta \rhd_{\tilde{a}_N} \mathfrak{C}_a \mid (\mathsf{new}\,\Phi_N)\,N \mid \lambda[\![\omega\,!\,\langle V_\Omega\rangle]\!] \Downarrow \delta$. Les contextes $\mathfrak{C}_a$ permettent donc de prouver que $\Omega \rhd_{\tilde{a}_M} M \Downarrow a$ si et seulement si $\Omega \rhd_{\tilde{a}_N} N \Downarrow a$.

fermée par extensions loyales Cette preuve se décompose en deux étapes. La première étape consistera à prouver qu'un processus peut engendrer

une extension loyale par une succession de newloc, newchan et newpass. La seconde étape consistera à prouver que le rajout d'un tel processus donne un système bisimilaire à celui où ces nouveaux noms sont déjà générés.

Construisons donc un contexte $\mathfrak{C}_e$ générant l'extension loyale. Ce contexte parallèle doit commencer par récupérer l'environnement Ω émis sous forme de valeur sur le canal ω. Il est donc de la forme $\lambda[\![\omega\,?\,(X : \mathsf{T}_\Omega)\,P]\!]$. On notera x_a pour la variable du motif X qui est associée au nom a dans la valeur V_Ω. L'objectif, en dernier ressort, du processus P est de transmettre le nouvel environnement $\Omega;\Phi$ où Φ est une extension loyale de Ω, donc il s'achèvera par le simple processus effectuant une émission $P_f = \omega_e\,!\,\langle V_{\Omega;\Phi}\{^X/_{V_\Omega}\}\rangle$. On note Φ sous la forme $a_1 : \mathsf{E}_1; \ldots; a_n : \mathsf{E}_n$. En prenant $P_{n+1} = P_f$ et en supposant que l'on ait déjà décrit le processus P_{i+1} permettant de générer les noms $a_{i+1}, \ldots, a_n$, on raisonne sur le type E_i pour générer a_i :

loc On génère simultanément la localité a_i et un passeport p_i autorisant l'accès depuis λ.

$$\begin{array}{rcl} P_i & = & \mathsf{newloc}\, a_i, (), (p_i), () : \mathsf{L}\ \mathsf{with\ stop\ in}\ P_{i+1} \\ \mathsf{L} & = & \sum x : \textsc{loc}.\ \sum y : \textsc{loc}.\ (), (\lambda \mapsto y), () \end{array}$$

p_i servira à retourner dans a_i ultérieurement pour générer d'éventuels canaux et passeports.

$\mathsf{C}_{@}a_j$ Bien entendu, il faut que j soit inférieur à i. On sait alors que la génération de la localité a_j a dû être accompagnée de la génération d'un passeport p_j.

$$P_i = \mathsf{goto}_{p_j}\, a_j.\ \mathsf{newchan}\, a_i : \mathsf{C}\ \mathsf{in}\ \mathsf{goto}_\pi\, \lambda.\ P_{i+1}$$

$\mathsf{C}_{@}l_m$ **où** $l_m \notin \mathbf{dom}(\Phi)$ l_m doit alors être dans le domaine de Ω donc, par définition des extensions loyales, elle doit y être accessible. Ω contient donc des suites de passeports $q_1, \ldots, q_m$ et de localité $l_1, \ldots, l_m$ qui prouvent cette accessibilité.

$$P_i = \mathsf{goto}_{x_{q_1}}\, x_{l_1}.\ \ldots \mathsf{goto}_{x_{q_m}}\, x_{l_m}.\ \mathsf{newchan}\, a_i : \mathsf{C}\ \mathsf{in}\ \mathsf{goto}_\pi\, \lambda.\ P_{i+1}$$

$\tilde{l}^\star \mapsto l$ Que l soit dans $\mathsf{dom}(\Omega)$ ou $\mathsf{dom}(\Phi)$, ce cas se traite exactement comme l'un des deux cas précédents pour accéder à la localité l.

On voit aisément que $\mathfrak{C}_e$ ainsi construit est bien typé dans l'environnement $\lambda\pi\omega;\omega_e$: $\textsc{rw}\langle \mathsf{T}_{\Omega;\Phi}\rangle_{@}\lambda$. Par fermeture par extension loyale et contexte parallèle on peut donc conclure :

$$\begin{array}{l} \lambda\pi\omega\omega_e \vDash \\ \quad (\mathsf{new}\,\Phi_M)(M \mid \lambda[\![\omega\,!\,\langle V_\Omega\rangle]\!]) \mid \mathfrak{C}_e \ {}_{\tilde{a}_M}{\cong^p}_{\tilde{a}_N}\ (\mathsf{new}\,\Phi_N)(N \mid \lambda[\![\omega\,!\,\langle V_\Omega\rangle]\!]) \mid \mathfrak{C}_e \end{array}$$

soit, par renforcement (lemme 4.57) et extension des portées,

$$\begin{array}{l} \lambda\pi\omega_e \vDash \\ \quad (\mathsf{new}\,\Phi_M)(M \mid \lambda[\![\omega\,!\,\langle V_\Omega\rangle]\!] \mid \mathfrak{C}_e) \ {}_{\tilde{a}_M}{\cong^p}_{\tilde{a}_N}\ (\mathsf{new}\,\Phi_N)(N \mid \lambda[\![\omega\,!\,\langle V_\Omega\rangle]\!] \mid \mathfrak{C}_e) \end{array}$$

De plus, par définition de $\mathfrak{C}_e$,

$$\lambda[\![\omega\,!\,\langle V_\Omega\rangle]\!] \mid \mathfrak{C}_e \Longrightarrow \lambda[\![\mathsf{stop}]\!] \mid a_{i_1}[\![\mathsf{stop}]\!] \mid \cdots \mid \lambda[\![\omega_e\,!\,\langle V_{\Omega;\Phi}\rangle]\!]$$

où les noms $a_{i_1}, \ldots$ sont les localités générées pour l'extension Φ. Cette remarque suffit à conclure que la relation $\mathcal{S}$ définie comme l'union de $\approx^{al}$ et de $\mathcal{S}^b$ avec

$$\lambda\pi\omega_e \vDash (\mathsf{new}\,\Phi_M)(M \mid \lambda[\![\omega\,!\,\langle V_\Omega\rangle]\!] \mid \mathfrak{C}_e)\ {}_{\tilde{a}_M}\mathcal{S}^b_{\tilde{a}_M}\ (\mathsf{new}\,\Phi_M)(M \mid \lambda[\![\omega_e\,!\,\langle V_{\Omega;\Phi}\rangle]\!])$$

est une bisimulation loyale à β près. En effet, si

$$\lambda\pi\omega_e \rhd_{\tilde{a}_M} (\mathsf{new}\,\Phi_M)(M \mid \lambda[\![\omega\,!\,\langle V_\Omega\rangle]\!] \mid \mathfrak{C}_e) \xrightarrow{\mu} \lambda\pi\omega_e\ \mathsf{après}\ \mu \rhd_{\tilde{a}''_M} M''$$

alors μ doit être τ. S'il s'agit de la communication sur le canal ω, une série de β-réductions permet d'atteindre

$$\lambda\pi\omega_e \rhd_{\tilde{a}_M} (\mathsf{new}\,\Phi_M)(M \mid \lambda[\![\mathsf{stop}]\!] \mid a_{i_1}[\![\mathsf{stop}]\!] \mid \cdots \mid \lambda[\![\omega_e\,!\,\langle V_{\Omega;\Phi}\rangle]\!])$$

qui est naturellement bisimilaire à $\lambda\pi\omega_e \rhd_{\tilde{a}_M} (\mathsf{new}\,\Phi_M)(M \mid \lambda[\![\omega_e\,!\,\langle V_{\Omega;\Phi}\rangle]\!])$ donc on obtient deux systèmes en relation par $\mathcal{S}$. S'il ne s'agit pas de la communication sur le canal ω, seul M peut se réduire pour devenir M'. Cette action est immédiatement copiable par l'autre système pour obtenir

$$\lambda\pi\omega_e \vDash (\mathsf{new}\,\Phi_M)(M' \mid \lambda[\![\omega\,!\,\langle V_\Omega\rangle]\!] \mid \mathfrak{C}_e)\ {}_{\tilde{a}_M}\mathcal{S}_{\tilde{a}_M}\ (\mathsf{new}\,\Phi_M)(M' \mid \lambda[\![\omega_e\,!\,\langle V_{\Omega;\Phi}\rangle]\!])$$

Si

$$\lambda\pi\omega_e \rhd_{\tilde{a}_M} (\mathsf{new}\,\Phi_M)(M \mid \lambda[\![\omega_e\,!\,\langle V_{\Omega;\Phi}\rangle]\!]) \xrightarrow{\mu} \lambda\pi\omega_e\ \mathsf{après}\ \mu \rhd_{\tilde{a}''_M} M''$$

alors μ peut être soit τ comme dans le cas précédent soit l'action $(\tilde{b})\omega_e!V_{\Omega;\Phi}$. Dans le premier cas, où μ est τ, seul M agit, l'action est donc immédiatement copiée par l'autre système. Dans le second, le résultat vient de

$$\lambda\pi\omega_e \rhd_{\tilde{a}_M} (\mathsf{new}\,\Phi_M)(M \mid \lambda[\![\omega\,!\,\langle V_\Omega\rangle]\!] \mid \mathfrak{C}_e) \xrightarrow{(\tilde{b})\omega_e!V_{\Omega;\Phi}} \lambda\pi\omega_e\ \mathsf{après}\ \mu \rhd_{\tilde{a}''_M} (\mathsf{new}\,\Phi''_M)(M \mid \lambda[\![\mathsf{stop}]\!] \mid a_{i_1}[\![\mathsf{stop}]\!] \mid \cdots \mid \lambda[\![\mathsf{stop}]\!])$$

donc le système obtenu après cette action est bisimilaire à M''.
L'inclusion de $\approx^{al}$ dans $\cong^p$ et la transitivité de $\cong^p$ permettent de conclure que

$$\lambda\pi\omega_e \vDash (\mathsf{new}\,\Phi_M)(M \mid \lambda[\![\omega_e\,!\,\langle V_{\Omega;\Phi}\rangle]\!])\ {}_{\tilde{a}_M}\cong^p_{\tilde{a}_N}\ (\mathsf{new}\,\Phi_N)(N \mid \lambda[\![\omega_e\,!\,\langle V_{\Omega;\Phi}\rangle]\!])$$

donc $\Omega;\Phi \vDash M\ {}_{\tilde{a}'_M}\mathcal{R}_{\tilde{a}'_N}\ N$.

fermée par contextes parallèles Considérons le contexte $[\cdot] \mid k[\![P]\!]$ avec $\Omega \vdash k[\![P]\!]$ et $k \in \mathcal{A}_\Omega$. Formons alors le contexte

$$\mathfrak{C}_p = \lambda[\![\omega\,?\,(X : \mathsf{T}_\Omega)\,\omega_p\,!\,\langle X\rangle \mid \mathsf{goto}_{x_{q_1}}\,x_{l_1}.\ \ldots \mathsf{goto}_{x_{q_m}}\,x_k.\ P\{^X/_{V_\Omega}\}]\!]$$

où $q_1, \ldots, q_m$ et $l_1, \ldots, l_m = k$ sont les suites de passeports et localités qui prouvent que k est accessible dans Ω, et x_a la variable du motif X qui correspond au nom a de la valeur V_Ω.
Le jugement $\Omega \mid \vdash_k P$ implique $\lambda\pi\omega\omega_p;\langle X : \mathsf{T}_\Omega\rangle \circledcirc\lambda \mid \vdash_{x_k} P\{^X/_{V_\Omega}\}$ d'après le lemme 4.9 car $\langle X : \mathsf{T}_\Omega\rangle = \langle V_\Omega : \mathsf{T}_\Omega\rangle\,\{^X/_{V_\Omega}\}$. Par conséquent, $\lambda\pi\omega\omega_p \vdash \mathfrak{C}_p$.

Par fermeture par extension loyale et contexte parallèle on peut donc conclure :

$$\lambda\pi\omega\omega_p \vDash (\mathsf{new}\,\Phi_M)(M \mid \lambda[\![\omega\,!\,\langle V_\Omega\rangle]\!]) \mid \mathfrak{C}_p \;\tilde{a}_M{\cong^p}_{\tilde{a}_N}\; (\mathsf{new}\,\Phi_N)(N \mid \lambda[\![\omega\,!\,\langle V_\Omega\rangle]\!]) \mid \mathfrak{C}_p$$

soit, par renforcement (lemme 4.57) et extension des portées,

$$\lambda\pi\omega_p \vDash (\mathsf{new}\,\Phi_M)(M \mid \lambda[\![\omega\,!\,\langle V_\Omega\rangle]\!] \mid \mathfrak{C}_p) \;\tilde{a}_M{\cong^p}_{\tilde{a}_N}\; (\mathsf{new}\,\Phi_N)(N \mid \lambda[\![\omega\,!\,\langle V_\Omega\rangle]\!] \mid \mathfrak{C}_p)$$

En raisonnant de la même façon que dans le cas de fermeture par extension loyale, on obtient

$$\lambda\pi\omega_p \vDash (\mathsf{new}\,\Phi_M)(M \mid \lambda[\![\omega\,!\,\langle V_\Omega\rangle]\!] \mid \mathfrak{C}_p) \;\tilde{a}_M{\cong^p}_{\tilde{a}_M}\; (\mathsf{new}\,\Phi_M)(M \mid k[\![P]\!] \mid \lambda[\![\omega\,!\,\langle V_\Omega\rangle]\!])$$

ce qui achève de prouver que

$$\Omega \vDash M \mid k[\![P]\!] \;\tilde{a}'_M\mathcal{R}_{\tilde{a}'_N}\; N \mid k[\![P]\!]$$

et par conséquent de montrer aussi que $\mathcal{R}{\subseteq}{\cong^p}$.

□

Enfin, avant de prouver concrètement que $\cong^p$ est incluse dans $\approx^{al}$, on met en place un petit lemme, pendant de 4.37 pour la bisimulation.

Lemme 4.57 (Renforcement des environnements dans la congruence parallèle)**.** *Si* $\Omega \vDash M \;\tilde{a}_M{\cong^p}_{\tilde{a}_N}\; N$ *et* $\Omega <: \Omega'$ *alors* $\Omega' \vDash M \;\tilde{a}_M{\cong^p}_{\tilde{a}_N}\; N$.

Démonstration. La preuve fonctionne simplement en remarquant que la relation $\mathcal{R}$ définie par

$$\Omega' \vDash M \;\tilde{a}_M\mathcal{R}_{\tilde{a}_N}\; N$$

vérifie les critères de définition de $\cong^p$ et s'avère donc incluse dans celle-ci. Toutes les vérifications sont routinières puisque toutes les capacités requises de la part de Ω' sont en particulier disponibles dans Ω. □

Lemme 4.58 (La congruence parallèle typée barbue fermée par réduction et loyale est incluse dans la bisimilarité loyale)**.** $\cong^p \subseteq \approx^{al}$

Démonstration. Considérons $\Omega \vDash M \;\tilde{a}_M{\cong^p}_{\tilde{a}_N}\; N$. Par définition, $\cong^p$ est symétrique. Il suffit donc de vérifier que $\cong^p$ est une bisimulation loyale en considérant une action de M et en la simulant pour N.

Supposons $\Omega \rhd_{\tilde{a}_M} M \xrightarrow{\mu} \Omega$ après $\mu \rhd_{\tilde{a}'_M} M'$. On raisonne sur μ.

$\mu = \tau$ La coïncidence des sémantiques indique que $M \longrightarrow M'$. La fermeture par réductions de $\cong^p$ assure alors qu'il existe N' tel que $N \Longrightarrow N'$ avec $\Omega \vDash M' \;\tilde{a}_M{\cong^p}_{\tilde{a}_N}\; N'$. Toujours par coïncidence, on en déduit qu'il existe $N'' \equiv N$ tel que $\Omega \rhd_{\tilde{a}_N} N \stackrel{\hat{\tau}}{\Longrightarrow} \Omega \rhd_{\tilde{a}_N} N''$. Or nous avons déjà vu les preuves des inclusions suivantes : $\equiv_\iota \subseteq \approx^{al} \subseteq {\cong^{al}} \subseteq {\cong^p}$, ce qui assure que $\Omega \vDash M' \;\tilde{a}_M{\cong^p}_{\tilde{a}_N}\; N''$.

$\mu = (\tilde{b})c!V$ Considérons l'ensemble $\mathcal{N} = \mathsf{dom}(\Omega) \cup (\mathsf{fn}(M) \setminus \tilde{a}_M) \cup (\mathsf{fn}(N) \setminus \tilde{a}_N)$. Le lemme 4.52 donne :

$$M \mid \mathfrak{C}^{\Omega}_{\mathcal{N}}(\mu) \mid \mathfrak{C}_{fin} \Longrightarrow (\mathsf{new}\,\Phi_M)(M' \mid \mathfrak{C}_d \mid \lambda[\![\omega\,!\,\langle V_{\Omega \mathsf{après} \mu}\rangle]\!])$$

Par ailleurs, la fermeture par extension loyale et par contexte parallèle de $\cong^p$ assure que

$$\Omega\lambda\pi\omega\delta \vDash M \mid \mathfrak{C}^{\Omega}_{\mathcal{N}}(\mu) \mid \mathfrak{C}_{fin}\ {}_{\tilde{a}_M}\cong^p_{\tilde{a}_N}\ N \mid \mathfrak{C}^{\Omega}_{\mathcal{N}}(\mu) \mid \mathfrak{C}_{fin}$$

où

$$\Omega\lambda\pi\omega\delta = \Omega; \lambda : \textsc{loc}; \pi : \star \mapsto \lambda; \omega : \textsc{rw}\langle \mathsf{T}_{\Omega \mathsf{après} \alpha}\rangle_{@}\lambda; \delta : \textsc{rw}\langle\rangle_{@}\lambda$$

et la fermeture par réduction impose l'existence d'un N'' tel que

$$N \mid \mathfrak{C}^{\Omega}_{\mathcal{N}}(\mu) \mid \mathfrak{C}_{fin} \Longrightarrow N''$$

avec

$$\Omega\lambda\pi\omega\delta \vDash (\mathsf{new}\,\Phi_M)(M' \mid \mathfrak{C}_d \mid \lambda[\![\omega\,!\,\langle V_{\Omega \mathsf{après} \mu}\rangle]\!])\ {}_{\tilde{a}_M}\cong^p_{\tilde{a}_N}\ N''$$

On déduit du respect des barbes de $\cong^p$ que $\Omega\lambda\pi\omega\delta \rhd_{\tilde{a}_N} N'' \Downarrow \omega$ et $\Omega\lambda\pi\omega\delta \rhd_{\tilde{a}_N} N'' \not\Downarrow \delta$. Le lemme 4.52 achève donc de prouver que N'' est équivalent à un système $(\mathsf{new}\,\Phi_N)(N' \mid \mathfrak{C}_d \mid \lambda[\![\omega\,!\,\langle V_{\Omega \mathsf{après} \mu}\rangle]\!])$ avec $\Omega \rhd_{\tilde{a}_N} N \stackrel{\mu}{\Longrightarrow} \Omega\ \mathsf{après}\ \mu \rhd_{\tilde{a}'_N} N'$. Enfin, on déduit de l'hypothèse affaiblie par le lemme 4.57

$$\lambda\pi\omega\delta \vDash (\mathsf{new}\,\Phi_M)(M' \mid \mathfrak{C}_d \mid \lambda[\![\omega\,!\,\langle V_{\Omega \mathsf{après} \mu}\rangle]\!])\ {}_{\tilde{a}_M}\cong^p_{\tilde{a}_N} \\ {}_{\tilde{a}_M}\cong^p_{\tilde{a}_N}\ (\mathsf{new}\,\Phi_N)(N' \mid \mathfrak{C}_d \mid \lambda[\![\omega\,!\,\langle V_{\Omega \mathsf{après} \mu}\rangle]\!])$$

que

$$\lambda\pi\omega\delta \vDash \\ (\mathsf{new}\,\Phi_M)(M' \mid \lambda[\![\omega\,!\,\langle V_{\Omega \mathsf{après} \mu}\rangle]\!])\ {}_{\tilde{a}_M}\cong^p_{\tilde{a}_N}\ (\mathsf{new}\,\Phi_N)(N' \mid \lambda[\![\omega\,!\,\langle V_{\Omega \mathsf{après} \mu}\rangle]\!])$$

car $\approx^{al} \subseteq \cong^p$ puis que

$$\Omega\ \mathsf{après}\ \mu \vDash M'\ {}_{\tilde{a}'_M}\cong^p_{\tilde{a}'_N}\ N'$$

par le lemme 4.56, ce qui achève la preuve de ce cas.

$\mu = (\Phi)c?V$ La preuve de la transition $\Omega \rhd M \xrightarrow{\mu} \Omega; \Phi \rhd M'$ peut forcément se récrire, par commutations de la règle (TE-AFF) et par la proposition 4.21 de compositions des extensions loyales, sous la forme :

$$\frac{\Omega; \Phi \rhd M \xrightarrow{c?V} \Omega; \Phi \rhd M'}{\Omega \rhd M \xrightarrow{(\Phi)c?V} \Omega; \Phi \rhd M'}$$

$\Omega \vDash M\ {}_{\tilde{a}_M}\cong^p_{\tilde{a}_N}\ N$ implique, par fermeture par extension loyale et contexte parallèle :

$$\Omega; \Phi; \lambda\pi\delta\omega \vDash \\ M \mid \lambda[\![c\,!\,\langle V\rangle\,\omega\,!\,\langle V_{\Omega;\Phi}\rangle]\!] \mid \mathfrak{C}_{fin}\ {}_{\tilde{a}_M}\cong^p_{\tilde{a}_N}\ N \mid \lambda[\![c\,!\,\langle V\rangle\,\omega\,!\,\langle V_{\Omega;\Phi}\rangle]\!] \mid \mathfrak{C}_{fin}$$

Le lemme 4.55 implique alors que

$$M \mid \lambda[\![c\,!\,\langle V\rangle\,\omega\,!\,\langle V_{\Omega;\Phi}\rangle]\!] \mid \mathfrak{C}_{fin} \Longrightarrow M' \mid \mathfrak{C}_d \mid \lambda[\![\omega\,!\,\langle V_{\Omega;\Phi}\rangle]\!]$$

Alors la fermeture par réductions de $\cong^p$ impose l'existence d'une réduction $N \mid \lambda[\![c\,!\,\langle V\rangle\,\omega\,!\,\langle V_{\Omega;\Phi}\rangle]\!] \mid \mathfrak{C}_{fin} \Longrightarrow N''$ avec

$$\Omega;\Phi;\lambda\pi\delta\omega \vDash M' \mid \mathfrak{C}_d \mid \lambda[\![\omega\,!\,\langle V_{\Omega;\Phi}\rangle]\!] \ {}_{\tilde{a}_M}\cong^p_{\tilde{a}_N} N''$$

Le respect des barbes entraîne alors que $\Omega;\Phi;\lambda\pi\delta\omega \rhd_{\tilde{a}_N} N'' \Downarrow \omega$ et $\Omega;\Phi;\lambda\pi\delta\omega \rhd_{\tilde{a}_N} N'' \not\Downarrow \delta$. Le lemme 4.55 permet d'en déduire que N'' est de la forme $N' \mid \mathfrak{C}_d \mid \lambda[\![\omega\,!\,\langle V_{\Omega;\Phi}\rangle]\!]$ avec $\Omega \rhd_{\tilde{a}_N} N \xrightarrow{(\Phi)c?V_i} \Omega;\Phi \rhd_{\tilde{a}_N} N'$. Dans ce cas dégénéré, l'utilisation de l'inclusion de $\approx^{al}$ dans $\cong^p$, la transitivité de $\cong^p$ et le lemme 4.56 d'ouverture des portées permet de conclure cependant que

$$\Omega;\Phi \vDash M' \ {}_{\tilde{a}_M}\cong^p_{\tilde{a}_N} N'$$

ce qui achève la preuve.

□

Après ces longs développements, nous pouvons enfin conclure :

Théorème 4.59 (Coïncidence des équivalences)**.** *$\Omega \vDash M \cong^l N$ si et seulement si $\Omega \vDash M \ {}_{\emptyset}\approx^{al}_{\emptyset} N$.*

Démonstration. D'après les lemmes 4.44 et 4.58, $\cong^p = \approx^{al}$. Et d'après la proposition 4.32, $\Omega \vDash M \ {}_{\emptyset}\cong^{al}_{\emptyset} N$ si et seulement si $\Omega \vDash M \cong^l N$. □

4.8 Extension du domaine de contrôle

Comme annoncé dans la section 4.1 d'introduction du présent chapitre, on voudrait pouvoir utiliser une autre famille de passeports pour contrôler les accès aux ressources des processus après leur migration. Il existe sans doute de nombreuses façons distinctes pour mettre en place un tel contrôle. On ne fera ici qu'esquisser une approche de cette question.

Illustrons par un exemple l'approche proposée. Cet exemple fournit toutes les idées intéressantes d'une formalisation générale. Supposons qu'une localité l accepte de recevoir un processus uniquement à la condition qu'il n'y utilise aucune autre ressource que la capacité d'émission sur un canal *info*. Si p est le passeport qui régit cet accès à la localité, on peut imaginer de décrire un processus tentant d'utiliser cette ressource de la façon suivante :

$$\mathsf{goto}_p\, l\, (x_{info} : \mathrm{R}\langle \mathsf{T}\rangle)\, P$$

où la continuation qui s'exécutera à proprement parler dans la localité l est paramétrée par une variable x_{info} qui sera instanciée à l'instant où le processus entre. Dès lors, le système

$$k[\![\mathsf{goto}_p\, l\, (x_{info} : \mathrm{R}\langle \mathsf{T}\rangle)\, P]\!]$$

doit pouvoir se réduire en

$$l[\![P\{^{info}/_{x_{info}}\}]\!]$$

où le canal *info* qui instancie le paramètre est dicté par le passeport p. Pour que cette réduction soit possible, il est indispensable de disposer, à l'instant de la réduction, de l'argument associé au passeport utilisé. La réduction $\longrightarrow$ d'un système dépendra d'un environnement φ associant à chaque passeport libre existant dans le système son argument. φ aura pour cela la forme $\ldots, p : \mathit{info}, \ldots$ et la réduction sera notée $\longrightarrow_\varphi$. En somme :

$$k[\![\mathsf{goto}_p\, l\, (x_{\mathit{info}} : \mathrm{R}\langle \mathsf{T}\rangle)\, P]\!] \longrightarrow_{\ldots, p:\mathit{info}, \ldots} l[\![P\{^{\mathit{info}}/_{x_{\mathit{info}}}\}]\!]$$

Bien entendu, il faut absolument que l'argument correspondant à un passeport donné soit fixé à la création dudit passeport. Intuitivement, c'est en effet cette primitive qui modifie la politique de la localité, en déclarant que désormais les processus invoquant ce nouveau passeport auront le droit d'y entrer, et l'argument auquel ces processus auront accès est partie intégrante de ce droit. La primitive de création de ce passeport prendra par conséquent la forme

$$\mathsf{newpass}\, p\, \mathsf{from}\, k\, \mathsf{with}\, \mathit{info} : \mathrm{R}\langle \mathsf{T}\rangle\, \mathsf{in}\, Q$$

On définit la réduction de cette primitive de la façon suivante :

$$l[\![\mathsf{newpass}\, p\, \mathsf{from}\, k\, \mathsf{with}\, \mathit{info} : \mathrm{R}\langle \mathsf{T}\rangle\, \mathsf{in}\, Q]\!] \longrightarrow_\varphi (\mathsf{new}\, p_{\mathit{info}} : k \mapsto_{\mathrm{R}\langle \mathsf{T}\rangle} l)\, l[\![Q]\!]$$

c'est-à-dire que l'on annote le nom p par son argument, afin de pouvoir utiliser les règles de la congruence structurelle en considérant *info* libre dans ce new. On peut remarquer que, dans cette réduction, φ ne joue aucun rôle : les seules réductions où cet environnement intervient sont des migrations ; mais il apparaît naturellement dans toutes les autres réductions.

Considérons comment la migration peut être effectuée dans le système

$$(\mathsf{new}\, p_{\mathit{info}} : k \mapsto_{\mathrm{R}\langle \mathsf{T}\rangle} l)\, M \mid k[\![\mathsf{goto}_p\, l\, (x_{\mathit{info}} : \mathrm{R}\langle \mathsf{T}\rangle)\, P]\!]$$

Comme il se doit, la liste d'associations φ annotant la réduction de ce système ne pourra pas mentionner le passeport p. Pour traiter ce cas, la règle de réduction pour (R-C-NEW) est modifiée de la façon suivante dans le cas où il s'agit d'un passeport annoté :

$$\frac{N \longrightarrow_{\varphi; p:V} N'}{(\mathsf{new}\, p_V : \tilde{l}^\star \mapsto_{\mathsf{T}} l)\, N \longrightarrow_\varphi (\mathsf{new}\, p_V : \tilde{l}^\star \mapsto_{\mathsf{T}} l)\, N'}$$

Maintenant que les primitives modifiées par rapport au calcul présenté dans les sections précédentes de ce chapitre ont été détaillées, regardons comment elles pourraient être pertinemment typées. Le typage des générations de nouveaux passeports ne réservera a priori aucune grande surprise : il s'agit là du pan facile du contrôle d'accès. Ce typage pourra donc prendre la forme, toujours dans le cas particulier considéré :

$$\frac{\Gamma \vdash_\iota \mathit{info} : \mathrm{R}\langle \mathsf{T}\rangle \qquad \Gamma; p : k \mapsto_{\mathsf{T}} l \mid \vdash_\iota Q}{\Gamma \mid \vdash_\iota \mathsf{newpass}\, p\, \mathsf{from}\, k\, \mathsf{with}\, \mathit{info} : \mathrm{R}\langle \mathsf{T}\rangle\, \mathsf{in}\, Q}$$

Exactement de la même façon, le typage de new dans le système

$$(\mathsf{new}\, p_{\mathit{info}} : k \mapsto_{\mathrm{R}\langle \mathsf{T}\rangle} l)\, N$$

dans un environnement Γ consistera dans la vérification que l'argument *info* est effectivement de type $\mathrm{R}\langle\mathsf{T}\rangle$ dans la localité à laquelle le passeport p offre accès, à savoir l, suivie des autres vérifications habituelles sur N dans l'environnement $\Gamma; p : k \mapsto_{\mathrm{R}\langle\mathsf{T}\rangle} l$.

Par contre le typage d'un processus migrant est plus complexe. Si l'on se contentait en effet d'une règle

$$\frac{\Gamma \vdash_k p : k \mapsto_{\mathsf{T}} l \quad \Gamma, \langle x_{info} : \mathrm{R}\langle\mathsf{T}\rangle\rangle \,@\, l \mid \vdash_l P}{\Gamma \mid \vdash_k \mathsf{goto}_p\, l\, (x_{info} : \mathrm{R}\langle\mathsf{T}\rangle)\, P}$$

le processus P disposerait, après sa migration, de tous les droits accordés via le passeport p, c'est-à-dire $\langle x_{info} : \mathrm{R}\langle\mathsf{T}\rangle\rangle \,@\, l$, *plus* tous les droits qui sont nécessaires pour typer le reste du système, à savoir Γ. Il est par conséquent indispensable d'effectuer une coupe claire dans l'environnement de typage dans lequel P est typé. L'idée simple que l'on propose pour cela est de supprimer de l'environnement de typage tout canal, soit :

$$\frac{\Gamma \equiv \Gamma_{lp}; \Gamma_c \quad \Gamma \vdash_k p : k \mapsto_{\mathsf{T}} l \quad \Gamma_{lp}, \langle x_{info} : \mathrm{R}\langle\mathsf{T}\rangle\rangle \,@\, l \mid \vdash_l P}{\Gamma \mid \vdash_k \mathsf{goto}_p\, l\, (x_{info} : \mathrm{R}\langle\mathsf{T}\rangle)\, P}$$

où Γ_{lp} est un environnement ne comportant que des localités et des passeports et Γ_c que des canaux. De cette façon, on assure simplement que P ne pourra pas utiliser d'autre canal que x_{info} tant qu'il n'aura pas migré. En particulier, la localité l protège ainsi tous ses autres canaux.

En l'état, ce typage comporte une faille de sécurité de taille : rien n'empêche P, une fois entré dans l, de créer un nouveau passeport pour y accéder, par exemple un passeport ne contrôlant pas les ressources à l'instant de la migration. Là encore, le choix que l'on propose est celui de la simplicité : puisqu'un processus entrant dans une localité en utilisant un tel passeport n'est pas, a priori, digne de confiance, le typage lui interdira toute forme de modification de la politique de la localité. La règle prendra finalement la forme

$$\frac{\Gamma \equiv \Gamma_{lp}; \Gamma_c \quad \Gamma \vdash_k p : k \mapsto_{\mathsf{T}} l \quad \Gamma_{lp}, \langle x_{info} : \mathrm{R}\langle\mathsf{T}\rangle\rangle \,@\, l \mid \vdash'_l P}{\Gamma \mid \vdash_k \mathsf{goto}_p\, l\, (x_{info} : \mathrm{R}\langle\mathsf{T}\rangle)\, P}$$

où le typage $\vdash'_u$ est défini de façon très similaire à $\vdash_u$, c'est-à-dire par exactement les mêmes règles au « prime » près sauf pour :

- les règles de typage des primitives $\mathsf{newpass}$ ne sont pas reproduites, pour interdire effectivement la génération de nouveaux passeports ;
- les deux règles de typage des primitives goto ont exactement les mêmes hypothèses dans les deux typages : quel que soit le typage, si le passeport utilisé par P est sans contrôle de ressource, la continuation sera typée avec $\vdash$, sinon ce sera avec $\vdash'$.

La raison pour traiter les goto comme dans le typage $\vdash$ est de ne pas obliger une localité donnée à assumer la vérification du comportement d'un processus après sa ressortie, car cela ne relève plus de ses compétences.

Regardons donc un exemple un soupçon plus complexe. Supposons un serveur qui attend des requêtes sur un canal *req* et qui autorise pour cela les processus à provenir de n'importe quelle localité avec un passeport p associé à l'argument *req* et de type $\star \mapsto_{\mathrm{W}\langle \mathsf{T}_{req}\rangle} sv$. Le type du passeport et son argument implique que les processus migrant dans la localité *sv* du serveur obtiennent en tout et pour tout la capacité d'émission sur le canal *req* du serveur. Ce serveur jouet pourra donc prendre la forme

$$sv[\![req\,?\,((x_s, x_c), x_p : \mathsf{T}_{req}) \cdots \mathsf{if}\ x_s = sv\ \mathsf{then}\ \mathsf{goto}_{x_p}\ x_c\,(x)\,x\,!\,\langle\rangle\ \ \mathsf{else}\ \ \mathsf{stop}]\!]$$

où T_{req} est le type dépendant $\sum x, y : \vec{\text{LOC}}.\ x \mapsto_{\mathrm{W}\langle\rangle} y$. Ce serveur effectue lui les opérations suivantes :

- réception d'une requête,
- traitement de cette requête,
- migration vers la localité spécifiée dans la requête avec le passeport correspondant, non sans avoir vérifié que l'origine du passeport était bien la localité *sv* du serveur lui-même[4],
- enfin émission de la réponse (vide, ici) sur le canal *rép* qui aura remplacé la variable x à l'exécution.

Un client pourra donc se présenter ainsi

$$cl[\![\mathsf{newchan}\ rép : \mathrm{RW}\langle\rangle\ \mathsf{in}\\ rép\,?\,()\,P \mid \mathsf{newpass}\,p_{cl}\,\mathsf{from}\,sv\,\mathsf{with}\,rép : \mathrm{W}\langle\rangle\,\mathsf{in}\,\mathsf{goto}_p\,sv\,(x)\,x\,!\,\langle(sv, cl), p_{cl}\rangle]\!]$$

c'est-à-dire effectuer, successivement :

- la génération d'un nouveau canal *rép* pour obtenir une réponse du serveur,
- sa scission en deux processus pour attendre la réponse du serveur dans l'un tandis que l'autre processus :
  - génère un nouveau passeport autorisant un processus en provenance de *sv* à entrer et émettre un message sur *rép*,
  - migre dans *sv*,
  - et y poste sa requête, composée de son nouveau passeport et des origine et cible de ce passeport, sur le canal *req*, qui substituera à l'exécution la variable x.

On se convainc facilement que ces systèmes sont bien typés dans un environnement constitué essentiellement des hypothèses

$$sv : \mathrm{LOC}, cl : \mathrm{LOC}, req : \mathrm{RW}\langle \mathsf{T}_{req}\rangle_{@sv}$$

Enfin, considérons brièvement la modification principale qu'il faut appliquer à la définition de congruence barbue pour prendre en compte ces passeports : quelles sont les observations possibles lorsque l'on dispose des connaissances et droits représentés par un environnement Ω ? Cette notion devra dépendre naturellement de φ. Une émission sur un canal a sera visible au sein d'une configuration $\Omega \rhd M$ dès que Ω possède un passeport permettant d'entrer dans la localité de a et que ce passeport permet d'obtenir la capacité de réception sur a. Formellement, on observera donc une barbe sur a dès que :

- $\Omega \vdash p : \star \mapsto_{\mathsf{T}} l$,

[4] La clôture totale des types transmis impose que les localités à partir desquelles la migration est autorisée soient explicitement mentionnées. Le serveur doit alors vérifier que la requête qu'il a reçue mentionne l'origine correcte.

- $a \in \mathsf{fn}(\varphi(p))$,
- si $\Omega \equiv \Omega_{lp}; \Omega_c$ où Ω_{lp} ne contient que des localités et des passeports et Ω_c uniquement des canaux, $\Omega_{lp}, \langle \varphi(p) : \mathsf{T} \rangle \circledcirc l \vdash_l a : \mathrm{R}\langle \mathsf{T}' \rangle$, pour un type T' quelconque,
- et il existe P, M' et Φ tels que

$$M \Longrightarrow \equiv (\mathsf{new}\,\Phi)(M' \mid l[\![a\,!\,\langle V \rangle\, P]\!])$$

avec $a, l \notin \mathsf{dom}(\Phi)$.

Bien entendu, les barbes de la définition 4.14 seront elles aussi toujours visibles. Comme pour le cas de la congruence loyale, cette notion minimale d'observable devrait suffire à détecter toutes les interactions par contextualité.

La complexité du modèle évoqué ci-dessus justifie le choix que l'on a fait d'étudier, dans un premier temps, la théorie du calcul dans lequel seuls les passeports sans contrôle sur les ressources sont utilisés.

Bibliographie

[AC93] R. M. Amadio & L. Cardelli – « Subtyping recursive types », *ACM Transactions on Programming Languages and Systems* **15** (1993), no. 4, p. 575–631.

[Bar84] H. P. Barendregt – *The lambda calculus : Its syntax and semantics*, revised éd., Elsevier/North-Holland, Amsterdam, London, New York, 1984.

[BH00] M. Berger & K. Honda – « The two-phase commitment protocol in an extended π-calculus », in *EXPRESS '00 : Proceedings of the 7th International Workshop on Expressiveness in Concurrency (State College, USA, August 21, 2000)* (L. Aceto & B. Victor, éds.), ENTCS, vol. 39.1, Elsevier Science Publishers, 2000.

[CG00] L. Cardelli & A. D. Gordon – « Mobile ambients », *Theoretical Computer Science* **240** (2000), no. 1, p. 177–213, An extended abstract appeared in *Proceedings of FoSSaCS '98* : 140–155.

[CHR05] A. Ciaffaglione, M. Hennessy & J. Rathke – « Proof methodologies for behavioural equivalence in DPI », in *FORTE* (F. Wang, éd.), Lecture Notes in Computer Science, vol. 3731, Springer, 2005, p. 335–350.

[Chu41] A. Church – *The calculi of lambda-conversion*, Annals of Mathematical Studies, no. 6, Princeton University Press, 1941.

[DM82] L. Damas & R. Milner – « Principal type-schemes for functional programs », in *Proceedings of the 9th ACM Symposium on Principles of Programming Languages* (Albuquerque), 1982, p. 207–212.

[FH05] A. Francalanza & M. Hennessy – « A theory of system behaviour in the presence of node and link failures », in *CONCUR* (M. Abadi & L. de Alfaro, éds.), Lecture Notes in Computer Science, vol. 3653, Springer, 2005, p. 368–382.

[GLP03] V. Gapeyev, M. Levin & B. Pierce – « Recursive subtyping revealed », *Journal of Functional Programming* **12** (2003), no. 6, p. 511–548, Une version préliminaire est parue dans les actes de l'*International Conference on Functional Programming (ICFP)*, 2000. Réédité comme chapitre 21 de [Pie02].

[HMR03] M. Hennessy, M. Merro & J. Rathke – « Towards a behavioural theory of access and mobility control in distributed systems », *Theoretical Computer Science* **322** (2003), p. 615–669.

[HR02] M. Hennessy & J. Riely – « Resource access control in systems of mobile agents », *Information and Computation* **173** (2002), p. 82–120.

[HRY05] M. Hennessy, J. Rathke & N. Yoshida – « SafeDpi : a language for controlling mobile code. », *Acta Informatica* **42** (2005), no. 4-5, p. 227–290.

[JM04] O. H. Jensen & R. Milner – « Bigraphs and mobile processes (revised) », Tech. Report UCAM-CL-TR-580, University of Cambridge, Computer Laboratory, 2004.

[JR04] A. Jeffrey & J. Rathke – « A theory of bisimulation for a fragment of concurrent ML with local names », *Theoretical Computer Science* **323** (2004), p. 1–48.

[LM87] K. G. Larsen & R. Milner – « A complete protocol verification using relativized bisimulation », in *Proceedings 14th ICALP,* Karlsruhe (T. Ottmann, éd.), Lecture Notes in Computer Science, vol. 267, Springer-Verlag, 1987, p. 126–135.

[Mil89] R. Milner – *Communication and concurrency*, International Series in Computer Science, Prentice Hall, 1989.

[Mil91] — , « The polyadic π-calculus : A tutorial », Tech. Report 91-180, Laboratory for Foundations of Computer Science, Department of Computer Science, University of Edinburgh, 1991.

[Mil99] — , *Communicating and mobile systems : the π-calculus*, Cambridge University Press, 1999.

[ML84] P. Martin-Löf – *Intuitionistic type theory*, Bibliopolis, Napoli, 1984.

[MPW92] R. Milner, J. Parrow & D. Walker – « A calculus of mobile processes, I + II », *Information and Computation* **100** (1992), p. 1–40, 41–77.

[MS92] R. Milner & D. Sangiorgi – « Barbed bisimulation », in *Proceedings ICALP '92* (Vienna), vol. 623, Springer-Verlag, 1992, p. 685–695.

[MV05] F. Martins & V. T. Vasconcelos – « History-based access control for distributed processes. », in *TGC* (R. D. Nicola & D. Sangiorgi, éds.), Lecture Notes in Computer Science, vol. 3705, Springer, 2005, p. 98–115.

[Nec97] G. C. Necula – « Proof-carrying code », in *Proceedings of the 24th ACM Symposium on Principles of Programming Languages* (Paris, France), 1997.

[Oak01] S. Oaks – *Java security*, second éd., O'Reilly & Associates, Inc., pub-ORA :adr, 2001, Covers JAAS and JSEE ; Writing and deploying secure applications ; Covers Java 1.1, Java 2, and JCE 1.2.1.

[Par81] D. Park – « Concurrency and automata on infinite sequences », in *Proceedings of the 5th GI-Conference on Theoretical Computer Science* (P. Deussen, éd.), Lecture Notes in Computer Science, vol. 104, Springer-Verlag, Berlin/New York, 1981, p. 167–183.

[Par87] J. Parrow – « Verifying a CSMA/CD-protocol with CCS », Tech. Report ECS-LFCS-87-18, Laboratory for Foundations of Computer Science, Edinburgh University, 1987, CSR-224-87.

[Pie02] B. C. Pierce – *Types and programming languages*, MIT Press, 2002.

[PS96] B. C. Pierce & D. Sangiorgi – « Typing and subtyping for mobile processes. », *Mathematical Structures in Computer Science* **6** (1996), no. 5, p. 409–453.

[RH03] J. Riely & M. Hennessy – « Trust and partial typing in open systems of mobile agents. », *J. Autom. Reasoning* **31** (2003), no. 3-4, p. 335–370.

[She18] M. Shelley – *Frankenstein, or The Modern Prometheus*, Lackington, Hughes, Harding, Mavor & Jones, 1818.

[SW01] D. Sangiorgi & D. Walker – *The π-calculus : a theory of mobile processes*, Cambridge University Press, 2001.

[Wal89] D. J. Walker – « Automated analysis of mutual exclusion algorithms using CCS », *Formal Aspects of Computing* **1** (1989), no. 3, p. 273–292.

Annexe A

Un peu de cambouis

Un petit chapelet de figures trop illisibles pour rester dans le corps du texte, et suffisamment connues pour ne pas y être indispensables.

Fig. A.1 Noms libres

$$\mathsf{fn}(a) = \{a\}$$
$$\mathsf{fn}(x) = \emptyset$$
$$\mathsf{fn}((V_1, \ldots, V_n)) = \bigcup_i \mathsf{fn}(V_i)$$

$$\mathsf{fn}(l[\![P]\!]) = \{l\} \cup \mathsf{fn}(P)$$
$$\mathsf{fn}(v\,!\,\langle V\rangle\, P) = \mathsf{fn}(v) \cup \mathsf{fn}(V) \cup \mathsf{fn}(P)$$
$$\mathsf{fn}(v\,?\,(X : \mathsf{T})\, P) = \mathsf{fn}(v) \cup \mathsf{fn}(\mathsf{T}) \cup \mathsf{fn}(P)$$
$$\mathsf{fn}(\mathsf{goto}\ v.\ P) = \mathsf{fn}(v) \cup \mathsf{fn}(P)$$
$$\mathsf{fn}(\mathsf{if}\ u_1 = u_2\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2) = \mathsf{fn}(u_1) \cup \mathsf{fn}(u_2) \cup \mathsf{fn}(P_1) \cup \mathsf{fn}(P_2)$$
$$\mathsf{fn}(\mathsf{newchan}\, c : \mathsf{C}\, \mathsf{in}\, P) = \mathsf{fn}(\mathsf{C}) \cup (\mathsf{fn}(P) \setminus \{c\})$$
$$\mathsf{fn}(\mathsf{newloc}\, k, (\vec{c}) : \mathsf{L}\, \mathsf{with}\, P_k\, \mathsf{in}\, P) = \mathsf{fn}(\mathsf{L}) \cup ((\mathsf{fn}(P_k) \cup \mathsf{fn}(P)) \setminus \{k, \tilde{c}\})$$
$$\mathsf{fn}(P_1 \mid P_2) = \mathsf{fn}(P_1) \cup \mathsf{fn}(P_2)$$
$$\mathsf{fn}(\mathsf{here}\,(x)\, P) = \mathsf{fn}(P)$$
$$\mathsf{fn}(*P) = \mathsf{fn}(P)$$
$$\mathsf{fn}((\mathsf{rec}\ Z : \mathsf{R}\,(X)\,.\ P)\,\langle V\rangle) = \mathsf{fn}(\mathsf{R}) \cup \mathsf{fn}(P) \cup \mathsf{fn}(V)$$
$$\mathsf{fn}(Z\,\langle V\rangle) = \mathsf{fn}(V)$$
$$\mathsf{fn}(\mathsf{stop}) = \emptyset$$

$$\mathsf{fn}(M_1 \mid M_2) = \mathsf{fn}(M_1) \cup \mathsf{fn}(M_2)$$
$$\mathsf{fn}((\mathsf{new}\, a : \mathsf{E})\, M) = \mathsf{fn}(\mathsf{E}) \cup (\mathsf{fn}(M) \setminus \{a\})$$
$$\mathsf{fn}(\mathbf{0}) = \emptyset$$

$$\mathsf{fn}(\mathrm{R}\langle \mathsf{T}\rangle) = \mathsf{fn}(\mathsf{T})$$
$$\mathsf{fn}(\mathrm{W}\langle \mathsf{T}\rangle) = \mathsf{fn}(\mathsf{T})$$
$$\mathsf{fn}(\mathrm{RW}\langle \mathsf{T}_1, \mathsf{T}_2\rangle) = \mathsf{fn}(\mathsf{T}_1) \cup \mathsf{fn}(\mathsf{T}_2)$$
$$\mathsf{fn}(\mathrm{LOC}) = \emptyset$$
$$\mathsf{fn}((\mathsf{T}_1, \ldots, \mathsf{T}_n)) = \bigcup_i \mathsf{fn}(\mathsf{T}_i)$$
$$\mathsf{fn}(\textstyle\sum \vec{x} : \vec{\mathrm{LOC}}.\ \mathsf{T}) = \mathsf{fn}(\mathsf{T})$$
$$\mathsf{fn}(\mathsf{C}_{@}s) = \mathsf{fn}(\mathsf{C}) \cup \mathsf{fn}(s)$$
$$\mathsf{fn}(\mathrm{REC}\,\mathbf{Y}.\ .\mathsf{T}) = \mathsf{fn}(\mathsf{T})$$
$$\mathsf{fn}(\mathbf{Y}) = \emptyset$$
$$\mathsf{fn}(\Gamma, s : \mathsf{E}) = \mathsf{fn}(\Gamma) \cup \mathsf{fn}(\mathsf{E})$$

Fig. A.2 Variables libres

$$\mathsf{fv}(a) = \emptyset$$
$$\mathsf{fv}(x) = \{x\}$$
$$\mathsf{fv}((V_1, \ldots, V_n)) = \bigcup_i \mathsf{fv}(V_i)$$

$$\mathsf{fv}(l[\![P]\!]) = \{l\} \cup \mathsf{fv}(P)$$
$$\mathsf{fv}(v\,!\,\langle V\rangle\, P) = \mathsf{fv}(v) \cup \mathsf{fv}(V) \cup \mathsf{fv}(P)$$
$$\mathsf{fv}(v\,?\,(X : \mathsf{T})\, P) = \mathsf{fv}(v) \cup \mathsf{fv}(\mathsf{T}) \cup (\mathsf{fv}(P) \setminus \mathsf{fv}(X))$$
$$\mathsf{fv}(\mathsf{goto}\ v.\ P) = \mathsf{fv}(v) \cup \mathsf{fv}(P)$$
$$\mathsf{fv}(\mathsf{if}\ u_1 = u_2\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2) = \mathsf{fv}(u_1) \cup \mathsf{fv}(u_2) \cup \mathsf{fv}(P_1) \cup \mathsf{fv}(P_2)$$
$$\mathsf{fv}(\mathsf{newchan}\ c : \mathsf{C}\ \mathsf{in}\ P) = \mathsf{fv}(\mathsf{C}) \cup \mathsf{fv}(P)$$
$$\mathsf{fv}(\mathsf{newloc}\ k, (\vec{c}) : \mathsf{L}\ \mathsf{with}\ P_k\ \mathsf{in}\ P) = \mathsf{fv}(\mathsf{L}) \cup \mathsf{fv}(P_k) \cup \mathsf{fv}(P)$$
$$\mathsf{fv}(P_1 \mid P_2) = \mathsf{fv}(P_1) \cup \mathsf{fv}(P_2)$$
$$\mathsf{fv}(\mathsf{here}\,(x)\, P) = \mathsf{fv}(P) \setminus \{x\}$$
$$\mathsf{fv}(*P) = \mathsf{fv}(P)$$
$$\mathsf{fv}((\mathsf{rec}\ Z : \mathsf{R}\,(X)\,.\ P)\,\langle V\rangle) = \mathsf{fv}(\mathsf{R}) \cup (\mathsf{fv}(P) \setminus \mathsf{fv}(X)) \cup \mathsf{fv}(V)$$
$$\mathsf{fv}(Z\,\langle V\rangle) = \mathsf{fv}(V)$$
$$\mathsf{fv}(\mathsf{stop}) = \emptyset$$

$$\mathsf{fv}(M_1 \mid M_2) = \mathsf{fv}(M_1) \cup \mathsf{fv}(M_2)$$
$$\mathsf{fv}((\mathsf{new}\ a : \mathsf{E})\, M) = \mathsf{fv}(M) \cup \mathsf{fv}(\mathsf{E})$$
$$\mathsf{fv}(\mathbf{0}) = \emptyset$$

$$\mathsf{fv}(\mathrm{R}\langle \mathsf{T}\rangle) = \mathsf{fv}(\mathsf{T})$$
$$\mathsf{fv}(\mathrm{W}\langle \mathsf{T}\rangle) = \mathsf{fv}(\mathsf{T})$$
$$\mathsf{fv}(\mathrm{RW}\langle \mathsf{T}_1, \mathsf{T}_2\rangle) = \mathsf{fv}(\mathsf{T}_1) \cup \mathsf{fv}(\mathsf{T}_2)$$
$$\mathsf{fv}(\mathrm{LOC}) = \emptyset$$
$$\mathsf{fv}((\mathsf{T}_1, \ldots, \mathsf{T}_n)) = \bigcup_i \mathsf{fv}(\mathsf{T}_i)$$
$$\mathsf{fv}(\textstyle\sum \vec{x} : \mathrm{L}\vec{\mathrm{O}}\mathrm{C}.\ \mathsf{T}) = \mathsf{fv}(\mathsf{T}) \setminus \tilde{x}$$
$$\mathsf{fv}(\mathsf{C}_{@s}) = \mathsf{fv}(\mathsf{C}) \cup \mathsf{fv}(s)$$
$$\mathsf{fv}(\mathrm{REC}\,\mathbf{Y}.\ .\mathsf{T}) = \mathsf{fv}(\mathsf{T})$$
$$\mathsf{fv}(\mathbf{Y}) = \emptyset$$
$$\mathsf{fv}(\Gamma, s : \mathsf{E}) = \mathsf{fv}(\Gamma) \cup \mathsf{fv}(s) \cup \mathsf{fv}(\mathsf{E})$$

Fig. A.3 Substitution, sans capture, des valeurs

$$
\begin{array}{l}
s_1\{^{s_2}/_{s_1}\} = s_2 \\
t\{^{s_2}/_{s_1}\} = t \text{ quand } t \neq s_1 \\
(V_1, \ldots, V_n)\{^{s_2}/_{s_1}\} = (V_1\{^{s_2}/_{s_1}\}, \ldots, V_n\{^{s_2}/_{s_1}\}) \\
(u\,!\,\langle V\rangle\,P)\{^{s_2}/_{s_1}\} = (u\{^{s_2}/_{s_1}\})\,!\,\langle V\{^{s_2}/_{s_1}\}\rangle\,P\{^{s_2}/_{s_1}\} \\
(u\,?\,(X : \mathsf{T})\,P)\{^{s_2}/_{s_1}\} = (u\{^{s_2}/_{s_1}\})\,?\,(X : \mathsf{T}\{^{s_2}/_{s_1}\})\,(P\{^{s_2}/_{s_1}\}) \text{ si } s_1, s_2 \notin \mathsf{fv}(X) \\
(\mathsf{goto}\ u.\ P)\{^{s_2}/_{s_1}\} = \mathsf{goto}\ (u\{^{s_2}/_{s_1}\}).\ (P\{^{s_2}/_{s_1}\}) \\
(\mathsf{if}\ u_1 = u_2\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2)\{^{s_2}/_{s_1}\} = \\
\qquad \mathsf{if}\ u_1\{^{s_2}/_{s_1}\} = u_2\{^{s_2}/_{s_1}\}\ \mathsf{then}\ P_1\{^{s_2}/_{s_1}\}\ \mathsf{else}\ P_2\{^{s_2}/_{s_1}\} \\
(\mathsf{newchan}\,c : \mathsf{C}\,\mathsf{in}\,P)\{^{s_2}/_{s_1}\} = \mathsf{newchan}\,c : \mathsf{C}\{^{s_2}/_{s_1}\}\,\mathsf{in}\,P\{^{s_2}/_{s_1}\} \text{ si } s_1 \neq c,\ s_2 \neq c \\
(\mathsf{newloc}\,k, (\vec{c}) : \mathsf{L}\,\mathsf{with}\,P_k\,\mathsf{in}\,P)\{^{s_2}/_{s_1}\} = \\
\qquad \mathsf{newloc}\,k, (\vec{c}) : \mathsf{L}\{^{s_2}/_{s_1}\}\,\mathsf{with}\,P_k\{^{s_2}/_{s_1}\}\,\mathsf{in}\,P\{^{s_2}/_{s_1}\} \\
\qquad\qquad \text{si } s_1 \neq k,\ s_2 \neq k \\
(P_1 \mid P_2)\{^{s_2}/_{s_1}\} = P_1\{^{s_2}/_{s_1}\} \mid P_2\{^{s_2}/_{s_1}\} \\
(\mathsf{here}\,(x)\,P)\{^{s_2}/_{s_1}\} = \mathsf{here}\,(x)\,P\{^{s_2}/_{s_1}\} \text{ quand } s_1 \neq x \\
(*P)\{^{s_2}/_{s_1}\} = *P\{^{s_2}/_{s_1}\} \\
((\mathsf{rec}\ Z : \mathsf{R}\,(X)\,.\,P)\,\langle V\rangle)\{^{s_2}/_{s_1}\} = (\mathsf{rec}\ Z : \mathsf{R}\{^{s_2}/_{s_1}\}\,(X)\,.\,P\{^{s_2}/_{s_1}\})\,\langle V\{^{s_2}/_{s_1}\}\rangle \\
\qquad\qquad \text{si } s_1, s_2 \notin \{Z\} \cup \mathsf{fv}(X) \\
(Z\,\langle V\rangle)\{^{s_2}/_{s_1}\} = Z\{^{s_2}/_{s_1}\}\,\langle V\{^{s_2}/_{s_1}\}\rangle \\
\mathsf{stop}\{^{s_2}/_{s_1}\} = \mathsf{stop}
\end{array}
$$

Fig. A.4 Substitution, sans capture, des appels récursifs

$$
\begin{array}{l}
((\mathsf{rec}\ Z' : \mathsf{R}'\,(X')\,.\,P)\,\langle V\rangle)[^{\mathsf{rec}\ Z:\mathsf{R}(X).\ Q}/_Z] = \\
\qquad\qquad (\mathsf{rec}\ Z' : \mathsf{R}'\,(X')\,.\,P[^{\mathsf{rec}\ Z:\mathsf{R}(X).\ Q}/_Z])\,\langle V\rangle \\
\qquad\qquad\quad \text{si } (\{Z'\} \cup \mathsf{fv}(X')) \cap (\mathsf{fid}(Q) \setminus \mathsf{fv}(X)) = \emptyset \\
(Z\,\langle V\rangle)[^{\mathsf{rec}\ Z:\mathsf{R}(X).\ Q}/_Z] = (\mathsf{rec}\ Z : \mathsf{R}\,(X)\,.\,Q)\,\langle V\rangle \\
(Z'\,\langle V\rangle)[^{\mathsf{rec}\ Z:\mathsf{R}(X).\ Q}/_Z] = Z'\,\langle V\rangle \text{ si } Z \neq Z' \\
\mathsf{stop}[^{\mathsf{rec}\ Z:\mathsf{R}(X).\ Q}/_Z] = \mathsf{stop}
\end{array}
$$

Dans tous les autres cas, la substitution sans capture passe directement sur la continuation ; par exemple :

$$
\begin{array}{l}
(u\,!\,\langle V\rangle\,P)[^{\mathsf{rec}\ Z:\mathsf{R}(X).\ Q}/_Z] = u\,!\,\langle V\rangle\,P[^{\mathsf{rec}\ Z:\mathsf{R}(X).\ Q}/_Z] \\
(u\,?\,(X' : \mathsf{T})\,P)[^{\mathsf{rec}\ Z:\mathsf{R}(X).\ Q}/_Z] = u\,?\,(X' : \mathsf{T})\,P[^{\mathsf{rec}\ Z:\mathsf{R}(X).\ Q}/_Z] \\
\quad \text{si } \mathsf{fv}(X') \cap (\mathsf{fv}(Q) \setminus \mathsf{fv}(X)) = \emptyset
\end{array}
$$

Table des figures

Résumé

Le calcul réparti est de plus en plus utilisé bien qu'il reste très mal maîtrisé. Cette thèse porte sur le Dπ-calcul, une extension simple du π-calcul dans laquelle tous les processus sont placés dans des localités afin de décrire leur répartition. Dans ce calcul, les processus peuvent communiquer localement et migrer entre localités. À côté des canaux de communication et des localités, on identifie une nouvelle famille d'identifiants, les passeports, permettant un contrôle fin des migrations de processus : un processus doit disposer d'un passeport adéquat pour entrer dans une localité.

Afin de structurer le calcul, on met en place un système de types qui associe un type à chaque identifiant pour vérifier qu'un processus n'utilise que les droits qu'il possède. L'ordre de sous-typage sur les types est étendu aux types de passeports suivant les localités d'origine des processus migrant. On démontre que cet ordre admet des bornes inférieures sous certaines conditions. On prouve également que les processus se conformant à cette politique de typage conservent cette propriété au cours de leurs réductions.

On étudie aussi l'équivalence observationnelle : quand des processus exhibent-ils des comportements indiscernables pour un observateur ? En présence de passeports, il est indispensable d'imposer à l'observateur d'être loyal, c'est-à-dire d'exiger la possession de passeports pour observer les communications ayant lieu dans les localités correspondantes. Ces contraintes définissent une congruence dite barbue loyale. On développe ensuite un système de transitions étiquetées tel que la bisimilarité loyale engendrée coïncide avec cette congruence barbue loyale.

Abstract

Distributed computation is increasingly used even though it is still only loosely controlled. This thesis deals with the Dπ-calculus, a simple extension of the π-calculus in which processes are located to describe distribution. In this calculus, processes can communicate locally and migrate between locations. Beside communication channels and locations, a new family of identifiers, passports, is added to provide a fine-grained control over process migrations : any process must own an appropriate passport to enter a location.

The calculus is structured through a type system which associate a type to every identifier and allows to check that a process uses only the rights it owns. The subtyping order over types is extended to passport types by considering the origin of migrating processes. The existence of greatest lower bounds is shown under some conditions. The fact that well-typed processes stay well-typed whenever they reduce is also proved.

Observational equivalence is also studied : when do process behave identically from the point of view of some observer ? In a calculus equipped with passports, it is mandatory to ask the observer to play fair, i.e. to require that it owns some passports to observe any behaviour in the corresponding locations. These constraints define a fair barbed congruence. A labelled transition system is subsequently developed so that the bisimilarity it generates coincides with the fair barbed congruence.